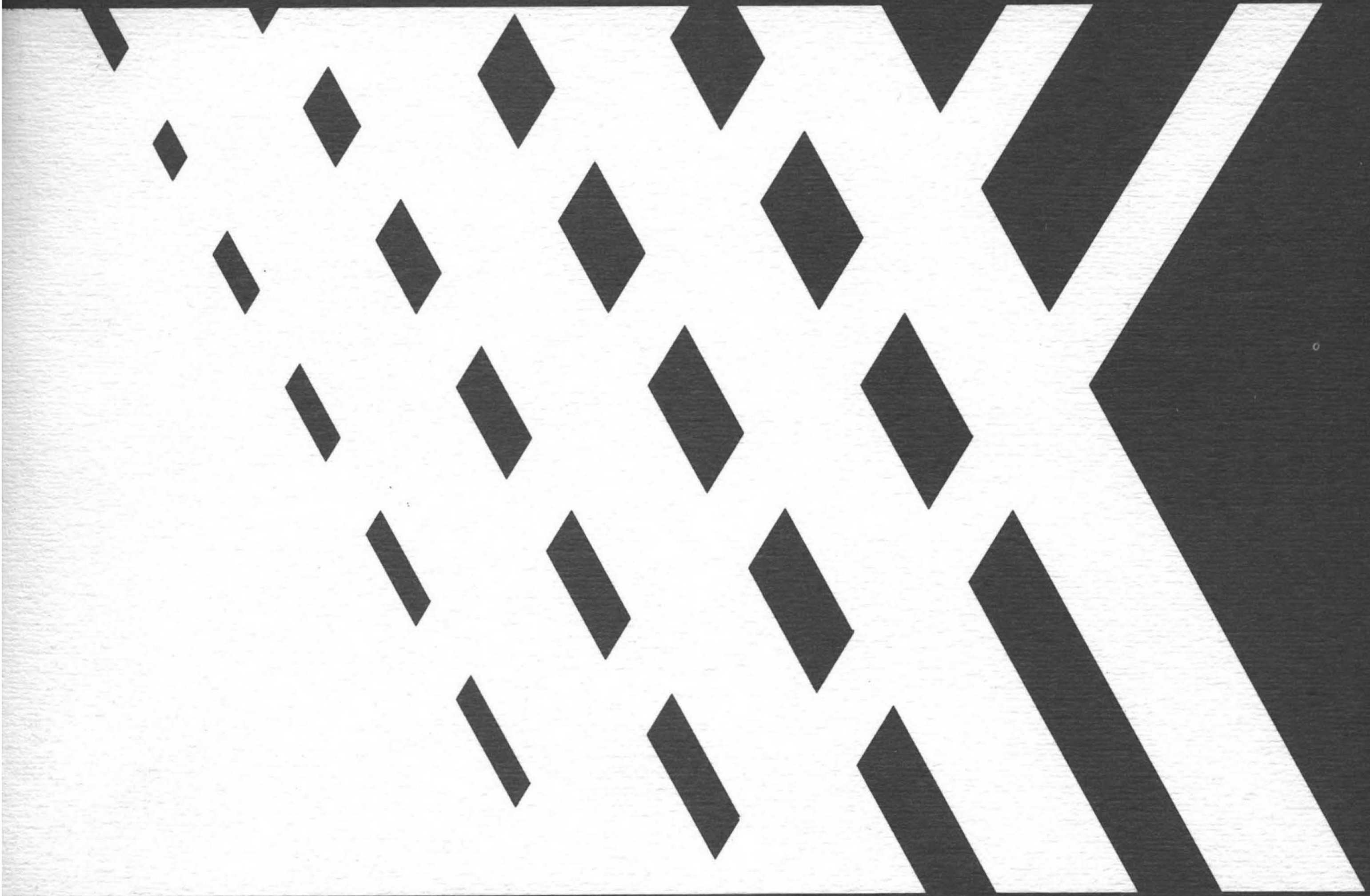
THE ANALYSIS OF HASHING ALGORITHMS

BY LEONIDAS J. GUIBAS



XEROX
PALO ALTO RESEARCH CENTER

THE ANALYSIS OF HASHING ALGORITHMS

BY LEONIDAS J. GUIBAS

CSL-76-3 JULY 1976

See abstract on next page.

KEY WORDS AND PHRASES

algorithmic analysis, arithmetic progression, hashing, information storage and retrieval, generating function, recurrence relation, Farey series, clustering, hypergeometric distribution, table overflow

CR CATEGORIES

3.74, 5.24, 5.25, 5.30

XEROX
PALO ALTO RESEARCH CENTER
3333 COYOTE HILL ROAD / PALO ALTO / CALIFORNIA 94304

ABSTRACT

In this thesis we relate the performance of hashing algorithms to the notion of *clustering*, that is the pile-up phenomenon that occurs because many keys may probe the table locations in the same sequence. We will say that a hashing technique exhibits *k-ary clustering* if the search for a key begins with k independent random probes and the subsequent sequence of probes is completely determined by the location of the k initial probes. Such techniques may be very bad; for instance, the average number of probes necessary for insertion may grow linearly with the table size. However, on the average (that is if the permutations describing the method are randomly chosen), k-ary clustering techniques for $k>1$ are very good. In fact the average performance is asymptotically equivalent to the performance of *uniform probing*, a method that exhibits no clustering and is known to be optimal in a certain sense.

Perhaps the most famous among tertiary clustering techniques is *double hashing*, the method in which we probe the hash table along arithmetic progressions where the initial element and the increment of the progression are chosen randomly and independently depending only on the key K of the search. We prove that double hashing is also asymptotically equivalent to uniform probing for load factors α not exceeding a certain constant $\alpha_0 = .31\ldots$. Our proof method has a different flavor from those previously used in algorithmic analysis. We begin by showing that the tail of the hypergeometric distribution a fixed percent away from the mean is exponentially small. We use this result to prove that random subsets of the finite ring of integers modulo m of cardinality αm have always nearly the expected number of arithmetic progressions of length k, except with exponentially small probability. We then use this theorem to start up a process (called the *extension process*) of looking at snapshots of the table as it fills up with double hashing. Between steps of the extension process we can show that the effect of clustering is negligible, and that we therefore never depart too far from the truly random situation.

TO MY PARENTS

ACKNOWLEDGEMENTS

This thesis is principally due to the encouragement of one man, my advisor Donald E. Knuth. He taught me how to do research, more by his general outlook towards problems than by specific guidance.

A lot of people contributed to the technical content of this thesis. The problems discussed in Sections 2.3 and 2.4 were suggested by Don Knuth, as well as the approach taken in Section 2.3. The entire analysis of double hashing (Chapter 3) is joint work with Endre Szemeredi. His insights into the frequency of "well-distributed configurations" were absolutely indispensible. The germ of the "pull-back process" idea of Section 3.4 is due to Vasek Chvatal.

A number of people have read all or parts of this dissertation and made valuable comments that helped improve the exposition and style. These include Luis Trabb-Pardo, Mark Brown, and Janet Roberts who also helped in the production of the final version of the dissertation.

General discussions with various individuals contibuted greatly to the content of this thesis. Among them are Ed McCreight, Paul Cohen, Robert Tarjan, Ben Wegbreit, Daniel Bobrow, Mike Paterson, and Peter Deutsch.

While a graduate student at Stanford University, the author was supported by the Hertz Foundation, the Xerox Palo Alto Research Center, the Danforth Foundation, and Stanford University.

Phyllis Winkler did a marvellous job of typing the original manuscript in record time. Carol VanJepmond produced the first machine readable form of this thesis. A lot of the more involved algebraic manipulation in this work, as well as the graphs displayed, were done using the MACSYMA system at MIT (supported by ARPA contract #N00014-75-C-0661). Finally thanks are due to the numerous people at Xerox PARC who provided the systems with which the current version of this thesis was produced, and who always volunteered to help when problems arose. Foremost among them are Charles Simonyi, Tom Malloy, Patrick Baudelaire, and William Newman.

NON-STANDARD NOTATIONS USED

$C(m,n)$ denotes the binomial coefficient that counts the number of ways to choose n things out of m without repetitions $= m!/n!(m-n)!$.

$C(m; n_1,n_2,\ldots,n_m)$, where $n_1+n_2+\ldots+n_m = m$, denotes the multinomial coefficient that equals $m!/n_1!n_2!\ldots n_m!$.

For positive x, y $x = y(1\pm\varepsilon)$ means $x \in \big(y(1-\varepsilon), y/(1-\varepsilon)\big)$ if $\varepsilon<1$ and $x < y(1+\varepsilon)$ otherwise.

Further notation is as in [Knuth1] or as defined in the text.

TABLE OF CONTENTS

THE ANALYSIS OF HASHING ALGORITHMS

Page

CHAPTER 1. INTRODUCTION 1

1.1 Hashing Algorithms 2

1.2 Open Address Versus Chained Hash Techniques 4

1.3 Algorithmic Analysis 8

1.4 Clustering 10

1.5 Some Sample Analyses 13

1.6 Summary of the Results 19

CHAPTER 2. RECURRENCE METHODS 21

2.1 Chained Scatter Searching with Overflow 22

2.2 A Combined Open Address and Chained Hash Search 29

2.3 The Worst k-ary Clustering Techniques 38

2.4 The Analysis Random k-ary Clustering Techniques 46

CHAPTER 3. THE ANALYSIS OF DOUBLE HASHING 55

3.1 The Lattice of Arithmetic Progressions Coming from a Set to a Point 56

3.2 The Tail of the Hypergeometric Distribution 61

3.3 The Farey Subdivision of the Circle 65

3.4 The Estimation of the Arithmetic Progressions, and the Prevalence of Randomness 70

3.5 Double Hashing 82

3.6 The Seed Set and the Final Argument 85

3.7 The Lattice Flows and the Extension Process 89

3.8 The Propagation of Errors and the Impotence of Clustering 96

APPENDIX 117

LIST OF ILLUSTRATIONS

Figure		Page
1.1.1	The hash function h as a mapping	3
1.4.1	The matrix for linear probing	12
1.4.2	The matrix for double hashing	12
1.5.1	Comparison of collision resolution methods	18
2.1.1	A chain with regular and overflow elements	25
2.2.1	Comparison of three algorithms	35
2.3.1	The Oracular Argument	47
2.4.1	The matrix defining a secondary clustering technique	49
3.1.1	The additive group Z_m	57
3.1.2	The lattice structure of the types of arithmetic progressions	58
3.1.3	An illustration of the "pull-back" argument	60
3.3.1	The subdivision into intervals of nonoverlapping multiples near a fixpoint	69
3.4.1	The pull-back process	73
3.7.1	The lattice of types of arithmetic progressions of a given length coming to a point	90
3.7.2	The mechanism of intertype flows	92
3.8.1	Camouflaging with the residual progessions	98
3.8.2	The good and bad insertions	100
3.8.3	The inflow and outflow of progressions from a type	107
3.8.4	The proliferation of δ's	116

CHAPTER 1:

INTRODUCTION

In this chapter we introduce the basic notions of hashing and of algorithmic analysis. We define terminology and notation to be used throughout this thesis. Finally we present a summary of the results to be proved.

1.1. Hashing Algorithms.

Hashing algorithms are a certain type of search procedure. We assume that we are given a set of *records*, where each record R is uniquely identified by its *key* K. Besides K the record R contains some unspecified useful information in the field INFO, as depicted in Figure 1.1.1.

We wish to organize our records in such a way that (1) we can quickly find the record having a given key K (if such a record exists), and (2) we can easily add additional records to our collection. Since all retrieval and update requests are specified exclusively in terms of the key of a record, we will ignore the INFO field in most of the discussion that follows. A straightforward way to implement this organization is to maintain our records in a table. A table entry is either *empty*, or it contains one of our records, in which case it is *full*. We can look for a record with a given key by exhaustively examining all entries of the table. Similarly, a new record can be inserted into the table by searching for an empty position. It is clear that, unless we are careful, the searches in question can become quite protracted for a large collection of records.

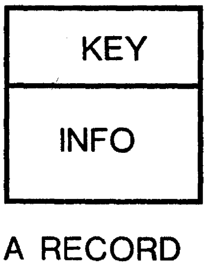
The idea of hashing is that of using a transformation h on the key K which gives us a "good guess" as to where in the table the record containing our key K is located. Suppose our table has m entries or positions, numbered 0,1,...,m-1. Then h maps the universe of keys, which we assume very large, into the set {0,1,...,m-1}. We call h a *hash function*, and depict it as a mapping, as in Figure 1.1.1.

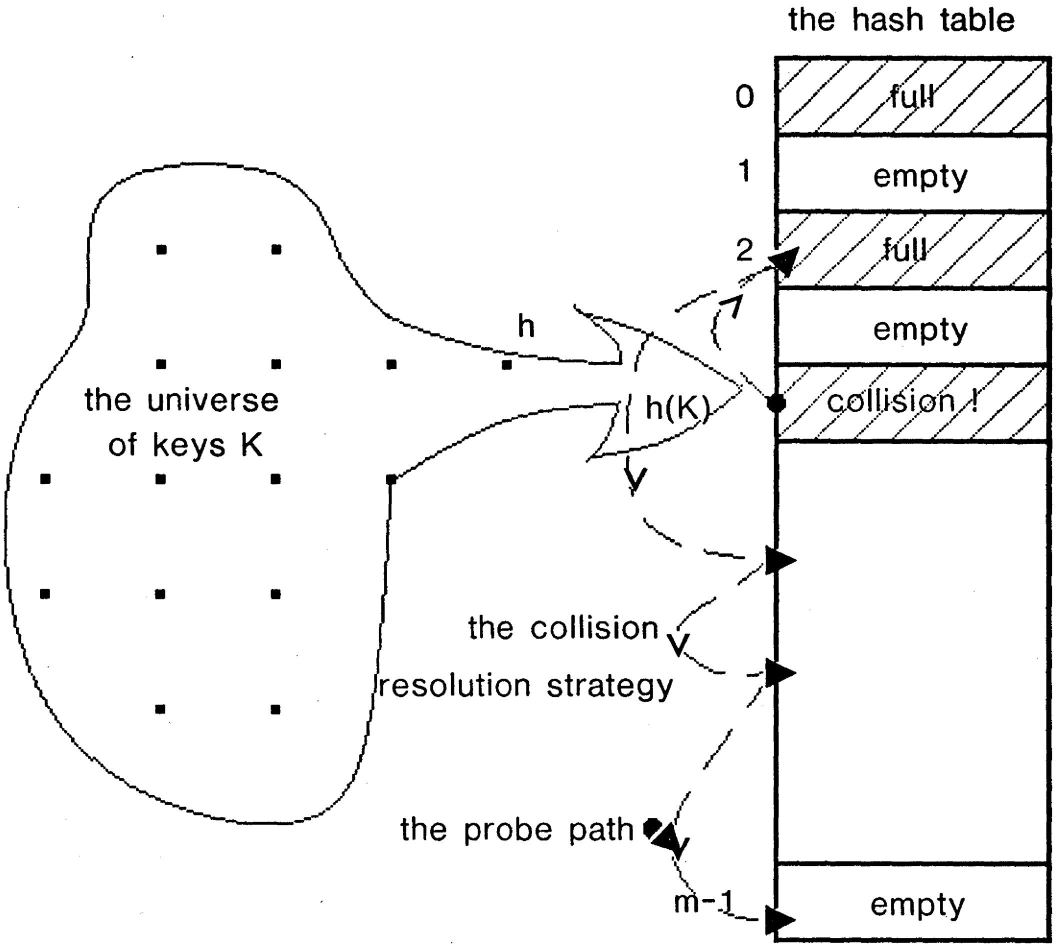
If h(K) = s, then we will say that key K *hashes to* position s. Naturally, several keys may hash to the same position. Thus if we are trying to insert a new key K into the table, it may happen that entry h(K) of the table is already occupied by another key. In that event we need a mechanism for probing the rest of the table until an empty entry is found. We will speak of a probe that encounters a full entry as a *collision*, and we will call our mechanism a *collision resolution strategy*. (It may, of course, happen that we are trying to insert a new key into an already full table, in which case we have an *overflow*.) Upon a retrieval request for the same key, we follow the same probe path until the record containing the key is found.

We will assume that our collision resolution strategy is such that every table position is examined exactly once before we return to the original location. The particular probe path we follow during a search may depend on the key K and the state of the table at that

Figure 1.1.1.

THE HASH FUNCTION h AS A MAPPING


KEY
INFO
A RECORD


the hash table
0
full
1
empty
2
full
empty
collision !
h
h(K)
the universe
of keys K
the collision
resolution strategy
the probe path
m-1
empty

moment, as the examples of the next section will make clear. We will also assume that our hash function selects each of the table entries with equal probability. It is intuitively clear that we want our function h to "randomly scatter" the keys over the entire table as much as possible. We will elaborate on these probabilistic concepts in Section 1.3. For the moment the point we wish to make is that, once the "uniformity" of h has been assumed, the collision resolution strategy alone fully determines the behavior of the algorithm. Thus every hashing algorithm we consider naturally breaks up into two parts: (1) the construction of the hash function h mapping the universe of possible keys into the set $\{0,1,\ldots,m-1\}$ so that each set member is chosen with approximately equal probability, and (2) the formulation of an efficient collision resolution strategy. Since in this thesis we are only concerned with the analysis of the performance of hashing algorithms, we will completely ignore the problem of constructing good hash functions. Similarly, if we use any additional randomizing transformations (hash functions) in the collision resolution strategy, we will only need to know the probability distribution of the values of such transformations. We will not concern ourselves with how such mappings can be explicitly constructed, given a specific universe of keys.

1.2. Open Address Versus Chained Hash Techniques.

A hashing algorithm is an *open addressing* method if the probe path we follow for a given key K depends only on this key. Thus each key determines a permutation of $\{0,1,\ldots,m-1\}$ which indicates the sequence in which the table positions are to be examined. Let n denote the number of records currently in the table. Perhaps the two best known open addressing hash algorithms are *linear probing* and *double hashing*. We use the descriptions of these algorithms given in [Knuth2]:

ALGORITHM L (Linear probing). This algorithm searches an m-node table, looking for a given key K. If K is not in the table and the table is not full, K is inserted.

The nodes of the table are denoted by TABLE[i], for $0 \leq i < m$, and they are of two distinguishable types, *empty* and *occupied*. An occupied node contains a key, called KEY[i], and possibly other fields. An auxiliary variable n is used to keep track of how many nodes are occupied; this variable is considered to be part of the table, and it is increased by 1 whenever a new key is inserted.

This algorithm makes use of a hash function h(K), and it uses a linear probing sequence to address the table.

L1. [Hash.] Set $i \leftarrow h(K)$. (Now $0 \leq i < m$.)

L2. [Compare.] If KEY[i] = K, the algorithm terminates successfully. Otherwise if TABLE[i] is empty, go to L4.

L3. [Advance to next.] Set $i \leftarrow i-1$; if now $i<0$, set $i \leftarrow i+m$. Go back to step L2.

L4. [Insert.] (The search was unsuccessful). If $n = m-1$, the algorithm terminates with overflow. (This algorithm considers the table to be full when $n = m-1$, not when $n = m$.) Otherwise set $n \leftarrow n+1$, mark TABLE[i] occupied, and set KEY[i] ← K. ▮

ALGORITHM D (Open addressing with double hashing). This algorithm is almost identical to Algorithm L, but it probes the table in a slightly different fashion by making use of two hash functions $h_1(K)$ and $h_2(K)$. As usual $h_1(K)$ produces a value between 0 and m-1, inclusive; but $h_2(K)$ must produce a value between 1 and m-1 that is *relatively prime* to m. (For example, if m is prime, $h_2(K)$ can be *any* value between 1 and m-1 inclusive; or if $m = 2^p$, $h_2(K)$ can be any *odd* value between 1 and 2^p-1.) The probe sequences in this case are arithmetic progressions.

D1. [First hash.] Set $i \leftarrow h_1(K)$.

D2. [First probe.] If TABLE[i] is empty, go to D6. Otherwise if KEY[i] = K, the algorithm terminates successfully.

D3. [Second hash.] Set $c \leftarrow h_2(K)$.

D4. [Advance to next.] Set $i \leftarrow i-c$; if now $i<0$, set $i \leftarrow i+m$.

D5. [Compare.] If TABLE[i] is empty, go to D6. Otherwise if KEY[i] = K, the algorithm terminates successfully. Otherwise go back to D4.

D6. [Insert.] If $n = m-1$, the algorithm terminates with overflow. Otherwsie set $n \leftarrow n+1$, mark TABLE[i] occupied, and set KEY[i] $\leftarrow$ K. ∎

We note that the main difference between these two algorithms is that in double hashing the decrement distance c can itself depend on the key K. As we will see later, this additional degree of freedom can have profound effects on the performance.

In non-open addressing methods the probe path of a key K depends on the previous history of the table. This is usually accomplished by storing in each record an additional LINK field which can be a pointer to another entry of the table. The probe path of a key is then determined by hashing to a location s and following the links from that location. If we hash to an empty entry or we come upon a null link, we know that the record we are looking for is not in the table. We will call such hash algorithms *chained.* Among the simplest and most widely used chained techniques are the following two:

ALGORITHM A (Bucket search). We assume that we have m list-heads HEAD[i], $0 \le i \le m-1$, each pointing to (a possibly empty) list of records. Each record has an additional LINK field that can be a pointer to another record, or null. The lists of the algorithm are kept disjoint. If a new record has $h(K) = s$, then this record is added to the end of the list pointed to by HEAD[s]. We also assume that we have an operation x <= AVAIL that makes x point to a block of memory where the new record can be stored.

A1. [Hash.] Set $i \leftarrow h(k)$.

A2. [Is there a list?] If HEAD[i] = null then let j <= AVAIL, set HEAD[i] $\leftarrow$ j and go to A5, else set i $\leftarrow$ HEAD[i].

A3. [Compare.] If K = KEY[i] the algorithm terminates successfully.

A4. [Advance to next.] If LINK[i] $\neq$ null then set i $\leftarrow$ LINK[i] and go back to A3, else let j <= AVAIL and set LINK[i] $\leftarrow$ j.

A5. [Insert.] Set KEY[j] $\leftarrow$ K, and LINK[j] $\leftarrow$ null.

ALGORITHM C (Chaining with coalescing lists). This algorithm searches an m-node table, looking for a given key K. If K is not in the table and the table is not full, K is inserted.

The nodes of the table are denoted by TABLE[i], for $0 \le i \le m$, and they are of two distinguishable types, *empty* and *occupied*. An occupied node contains a key field KEY[i], a link field LINK[i], and possibly other fields.

C1. [Hash.] Set $i \leftarrow h(K)+1$. (Now $1 \le i \le m$.)

C2. [Is there a list?] If TABLE[i] is empty, go to C6. (Otherwise TABLE[i] is occupied; we will look at the list of occupied nodes which starts here.)

C3. [Compare.] If $K = KEY[i]$, the algorithm terminates successfully.

C4. [Advance to next.] If $LINK[i] \neq null$, set $i \leftarrow LINK[i]$ and go back to step C3.

C5. [Find empty node.] (The search was unsuccessful, and we want to find an empty position in the table.) Decrease r one or more times until finding a value such that TABLE[r] is empty. If $r = 0$, the algorithm terminates with overflow (there are no empty nodes left); otherwise set $LINK[i] \leftarrow r$, $i \leftarrow r$.

C6. [Insert a new key.] Mark TABLE[i] as an occupied node, with $KEY[i] \leftarrow k$ and $LINK[i] \leftarrow null$. ▮

Chained methods require more storage because of the LINK fields but, as the analyses in [Knuth2] show, they usually outperform open addressing techniques with respect to number of probes for given m and n.

Note that all algorithms we have presented handle both lookup and insertion.

1.3. Algorithmic Analysis.

We are concerned with analyzing the performance of algorithms such as those presented in the previous section. A discussion of how the analysis of specific algorithms relates to computational complexity is given in [Knuth3]. We first have to define the cost measure by which we will evaluate the performance. The two usual cost measures are the space and time consumed by the algorithm. In the algorithms we consider, the space cost will be either fixed or trivially computable, except for one case which we analyze in detail. In order to make our time costs implementation independent we will use the number of probes made during a lookup as our basic cost function. This accounts, however, for only part of where the running time of a hashing algorithm is spent. The computation of the hash function(s) is another significant component. In comparing algorithms we cannot always factor this component out, as double hashing, for example, uses two hash function computations per search, vs. only one for linear probing. Having made this caveat we now strictly confine our attention to the number of probes made.

With any hash function it can happen that all the keys we insert will select the same probe sequence. In this unfortunate situation all the algorithms of the previous section reduce to a linear search of the table. Thus the worst case of hashing methods is not very interesting. We will be concerned with performance on the average. Before we can make precise the notion of the *average number of probes*, we need to specify the probability distribution of the inputs to our algorithms. We assume that every one of the hash functions we use will select each of its allowed values with equal probability, independently of all the others. Thus for Algorithms L, A, and C we will assume that $h(K) = s$ $(0 \leq s \leq m-1)$ with probability $1/m$. For double hashing we will take m to be prime and then assume that $(h_1(K), h_2(K)) = (i,j)$ with probability $1/m(m-1)$, for all (i,j) with $0 \leq i \leq m-1$, $1 \leq j \leq m-1$, $i \neq j$.

We now specify what we mean by the number of probes a bit more carefully. Let us consider the insertion of a new record. For open addressing techniques we will include in our count the very last probe in which an empty position was discovered. The other probes correspond to comparisons between keys. To avoid monotony of language we will use the terms probe and comparison interchangeably, even though this is misleading when it comes to the last probe. For chained techniques we will count one probe if we discover that the list is empty, and otherwise we will just use the number of list elements examined (i.e., in the case of insertion, the length of the list).

We clearly need to distinguish a successful from an unsuccessful search. We will measure the performance of a hashing algorithm by the following two quantities:

DEFINITION 1.3.1. Given any hashing algorithm we define C'_n to be the average number of probes made into the table when the (n+1)-st record is inserted (unsuccessful search). We include in this count the very last probe that discovered the empty position in an open addressing technique, and count one for discovering an empty list in the case of chained hashing. We assume all hash functions involved to choose each of their allowed values independently with equal probability.

Similarly, C_n will denote the average number of comparisons (or probes) made in a successful search using the algorithm, when the table contains n records. For C_n we assume that we are equally likely to look up any record present in the table.

In an open addressing technique it is clear that the number of comparisons required to look up a specific record is the same as the number of probes made when that record was inserted. This observation implies that

$$C_n = (1/n) \sum_{i=0}^{n-1} C'_i$$

Thus in open addressing C_n is just an average C'_n. For this reason C'_n will be the principal quantity we investigate for such algorithms.

The quantities C_n, C'_n naturally also depend on m, the table size. We will find that a convenient way to express the answers we seek is in terms of the *load* (or *occupancy*) *factor* α of the table, where $\alpha = n/m$. In several cases we will be unable to obtain C_n, C'_n as closed form expression of n, m. But in these cases we will still be able to obtain formulae for C'_n and C_n as functions of α (and possibly m) that are asymptotically valid. That is, as the table size m gets large, if the load factor α, $0<\alpha<1$, stays fixed, these functions of α will differ from the true values by errors of the order of $O(1/m)$, and which therefore rapidly decrease as m increases. In terms of the load factor we may write C_α, C'_α rather than C_n,

C'_n. In this "continuous" approximation the above relation between successful and unsuccessful searches for open addressing becomes

$$C_\alpha = (1/\alpha) \int_0^\alpha C'_\alpha \, d\alpha$$

We will have occasion to appreciate the power of this notation in the examples of Section 1.5., as well as in the following two chapters.

1.4. Clustering.

Since we are interested in the performance of hashing algorithms, we might ask the following question: what is the probability that two keys will follow exactly the same probe path? We can expect that the higher this probability, the more will different keys interfere with each other, and therefore the worse the performance of our algorithm will be. This interference phenomenon we will generally refer to as *clustering*. For example, in linear probing the probability that two keys will follow the same probe path is identical to the probability that they will hash to the same location, which is $1/m$. In double hashing this probability is easily seen to be $1/m(m-1)$. Thus we expect double hashing to have smaller C'_n (and C_n) than linear probing, as is indeed borne out by the analyses.

Another way to appreciate the effect of clustering is by observing that (loosely speaking) configurations of occupied positions that have a relatively high C'_n grow with a higher probability than configurations with a low C'_n. For example, in linear probing a long block of contiguous occupied positions gives us a large contribution to the total C'_n. During the next insertion the probability that such a block will grow by one is proportional to the length of the block. Thus long blocks grow into even longer ones with higher probability than short ones. This "pile-up" effect accounts for the rapid increase in C'_α for linear probing as $\alpha \rightarrow 1$. Similarly, in double hashing the configurations that contribute greatly to the mean C'_n are those that contain a large number of arithmetic progressions among the occupied positions. In general the probability that a given empty position will be filled during the current insertion is proportional to the number of arithmetic progressions coming from the occupied positions to that empty position. Here we have made the convention that we have $m-1$ arithmetic progressions of length 0, so as to properly account for the probability of hitting our position on the first probe. Thus in double hashing, sets of occupied entries with an excessive number of arithmetic progressions will tend to grow into sets with even more progressions.

The connection between clustering and C'_n leads us to introduce a new family of classes of hashing techniques, those that exhibit secondary, tertiary, and in general k-ary clustering ([Knuth2]). A hashing technique is said to exhibit *secondary* clustering, if the search into the table begins with *one* random probe, and then follows a fixed permutation which depends only on the location of this first probe. A hashing technique is said to exhibit *tertiary* clustering if it begins with *two* independently random probes into the table, and then probes the remaining table positions in a fixed permutation that can depend only on the locations of those first two probes. And in general a *k-ary* clustering technique begins the search in the table with k independent random probes and then continues along a permutation that depends on the locations of these first k probes only. (It is unfortunate that our terminology is somewhat inconsistent: secondary clustering is *1*-ary clustering, tertiary is *2*-ary; we have maintained the terms secondary and tertiary for historical reasons, while for the analyses in Chapter 2 the above meaning of k is the natural one.) Thus linear probing exhibits secondary clustering, whereas double hashing exhibits tertiary clustering. More formally, we can think of a secondary clustering technique as being specified by an m x (m-1) matrix, where we think of the rows of the matrix as indexed by $\{0,1,..,(m-1)\}$, and the row corresponding to i is a permutation of $\{0,1,...,(m-1)\} - \{i\}$ which specifies the order in which the remaining table positions are to be probed. Thus for linear probing we have the matrix depicted by Figure 1.4.1.

Similarly, a tertiary clustering technique is defined by an (m(m-1)) x (m-2) matrix, where we think of the rows as indexed by (i,j), $0 \leq i \neq j \leq m-1$ and row (i,j) specifies in which order to probe the remaining m-2 table positions when we make our first probe at i and our second probe at j. Thus the matrix corresponding to double hashing (assuming that m is prime) is as shown in Figure 1.4.2., where the rows specify the arithmetic progressions to be followed in the search.

It is convenient to introduce at this point an open addressing technique that exhibits "no clustering", namely *uniform hashing* (or *probing*). Uniform hashing has the property that after n keys have been inserted, all C(m,n) possible subsets of occupied positions are equally likely. To achieve this we first probe the table at $h_1(K)$, then at $h_2(K)$ where $h_2(K) \neq h_1(K)$, then at $h_3(K)$ where $h_3(K) \neq h_1(K), h_2(K)$, and so on. Here each h_i is assumed to select each of its allowed values with equal probability, independently of all the others. This method is certainly of no practical interest, since we have to compute arbitrarily many independent

Figure 1.4.1. THE MATRIX FOR LINEAR PROBING

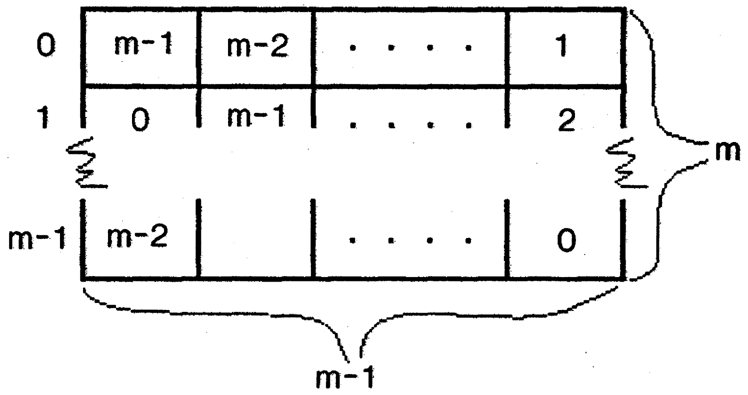
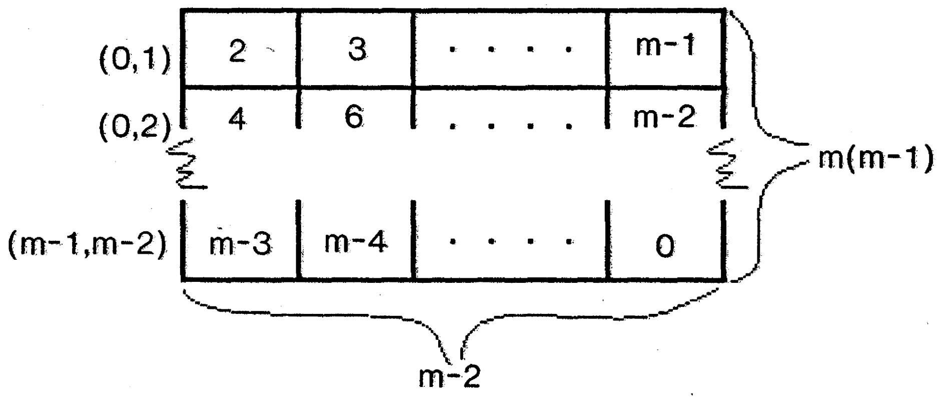

0
m-1
m-2
. . . . .
1
1
0
m-1
. . . . .
2
m
m-1
m-2
. . . . .
0
m-1

Figure 1.4.2. THE MATRIX FOR DOUBLE HASHING


(0,1)
2
3
. . . . .
m-1
(0,2)
4
6
. . . . .
m-2
m(m-1)
(m-1,m-2)
m-3
m-4
. . . . .
0
m-2

hash functions. On the other hand it is of theoretical importance, since Ullman has proved that no other open addressing technique can have a smaller C'_n for all n ([Ullman]). Thus the performance of uniform hashing, which we will compute in the next section, can be used as a benchmark against which to measure the success of other open addressing techniques.

The notion of clustering can also help us understand why we wisn to make our hash function h uniform, i.e., to make it equally likely to hash to any table entry. Suppose we are dealing with a technique with secondary clustering and let p_i denote the probability of hashing to entry i, $0 \leq i \leq m-1$. Then the probability that two keys will follow the same probe path is

$$\sum_{i=0}^{m-1} p^2_i$$

which, since $\Sigma_{0 \leq i < m}\ p_i = 1,$ is clearly minimized by setting

$$p_0 = p_1 = \dots = p_{m-1} = 1/m.$$

1.5. Some Sample Analyses.

For some simple hashing algorithms we can compute the average number of probes required for a successful or unsuccessful search directly, using only elementary combinatorial mathematics. In this section we will analyze uniform hashing and bucket search. We will also present without proofs the results of the analyses of linear probing and Algorithm C; detailed arguments can be found in [Knuth2]. These analyses will be a useful introduction to the techniques of the next two chapters. The results will also be useful for future reference.

We start with uniform hashing. Let p_{tr} denote the probability that we need r probes to insert the (t+1)-st element into the table. We have

$$C'_t = \sum_{1 \leq r \leq t} r\ p_{tr}.$$

It is often easier to compute this average another way, namely

$$C'_t = \sum_{1 \le r \le t} q_{tr},$$

where $q_{tr} = p_{tr} + p_{t,r+1} + \dots$ is the probability that *at least* r probes are needed. It is easily seen that, since all probes are random and independent,

$$q_{tr} = \frac{t\,(t-1)\,\dots\,(t-r+2)}{m\,(m-1)\dots(m-r+2)} = \frac{(m-r+1)!}{(m-t)!\,(t-r+1)!}\;\frac{(m-t)!\,t!}{m!} =$$

$$= C(m-r+1,m-t)/C(m,t).$$

Hence

$$C'_t = \sum_{1 \le r \le t} q_{tr} = 1/C(m,t) \sum_{1 \le r \le t} C(m-r+1,m-t) =$$

(by 1.2.6-(10) of [Knuth1]).

$$= C(m+1,m-t+1)/C(m,t) = 1 + t/(m-t+1).$$

Thus,

$$C'_n = 1 + n/(m-n+1).$$

For C_n we have

$$C_n = 1/n \sum_{0 \le t \le n-1} C'_t = (m+1)/n \sum_{0 \le t \le n-1} 1/(m-t+1)$$

$$= (m+1)/n \sum_{m-n+2 \le t \le m+1} 1/t = ((m+1)/n)\,(H_{m+1} - H_{m-n+1}).$$

Here H_i denotes the harmonic number $\Sigma_{1 \le k \le i}\ 1/k$. In terms of α, we can write these results as

$$C'_\alpha = 1/(1-\alpha) + O(1/m),$$

$$C_\alpha = (1/\alpha) \log 1/(1-\alpha) + O(1/m),$$

where in the latter we have used the fact that

$$H_i = \log i + \gamma + O(1/i),$$

with γ being Euler's constant ([Knuth1]). We see that the load factor has led to simple expressions. The relation

$$C_\alpha = 1/\alpha \int_0^\alpha C'_\alpha \, d\alpha$$

is evidently true.

Next we handle bucket search. We assume of course that all possible m^n input hash sequences are equally likely. Let p_{nk} denote the probability that a given list has length k. There are C(m,k) ways to choose the set of keys in our sequence that will hash to the given list, and $(m-1)^{n-k}$ ways to assign hash values to the other keys. Therefore

$$p_{nk} = C(n,k)\,(m-1)^{n-k}/m^n.$$

If we introduce the generating function ([Knuth1])

$$P_n(z) = \sum_{k=0}^{n} p_{nk}\, z^k,$$

then

$$P_n(z) = \sum_{k=0}^{n} C(n,k)\, ((m-1)^{n-k}/m^n) z^k = (1 + (z-1)/m)^n$$

by the binomial theorem. Note that

$$P'_n(1) = n/m, \quad P''_n(1) = n(n-1)/m^2.$$

We now have

$$C'_n = \sum_{k=0}^{n} (k+\delta_{k0}) p_{nk} = p'_n(1) + p_n(0).$$

counting one probe for discovering an empty chain

Thus

$$C'_n = n/m + (1-(1/m))^n.$$

For C_n consider the total number of probes to find all keys. A list of length k contributes C(k+1,2) to the total; hence

$$C_n = (m/n) \sum_{k=0}^{n} C(k+1,2)\; p_{nk} =$$

$$(m/n)(½\, P''_n(1) + P'_n(1)) = 1 + (n-1)/2m.$$

Asymptotically then we have

$$C'_\alpha = \alpha + e^{-\alpha} + O(1/m),$$

$$C_\alpha = 1 + \alpha/2 + O(1/m).$$

(Note that these results are valid also for $\alpha > 1$, since $n > m$ is possible in this method).

In Section 6.4 of [Knuth2] it is shown that Algorithm C leads to

$$C'_\alpha = 1 + (e^{2\alpha} - 1 - 2\alpha)/4 + O(1/m),$$

$$C_\alpha = 1 + (e^{2\alpha} - 1 - 2\alpha)/8\alpha + \alpha/4 + O(1/m),$$

which is only slightly worse than the performance of bucket search. The analysis of linear probing is a considerably more difficult problem. Knuth shows that

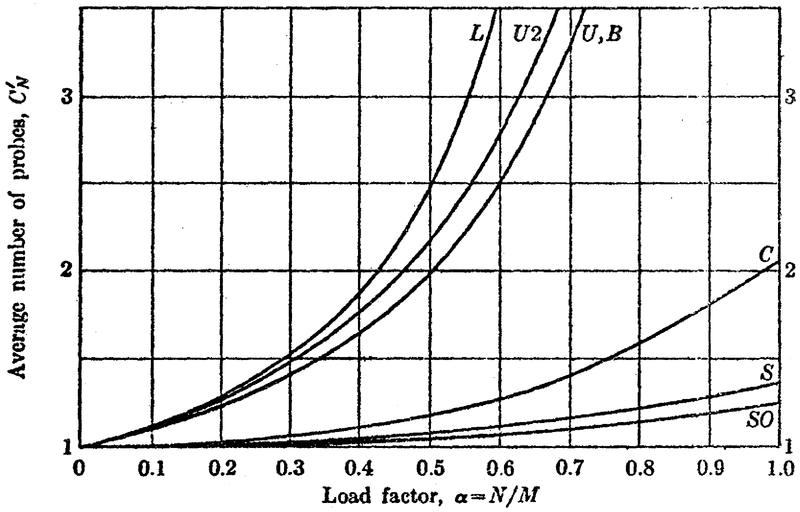
$$C'_\alpha = (1 + 1/(1-\alpha)^2)/2 + O(1/m),$$

$$C_\alpha = (1 + 1/(1-\alpha))/2 + O(1/m),$$

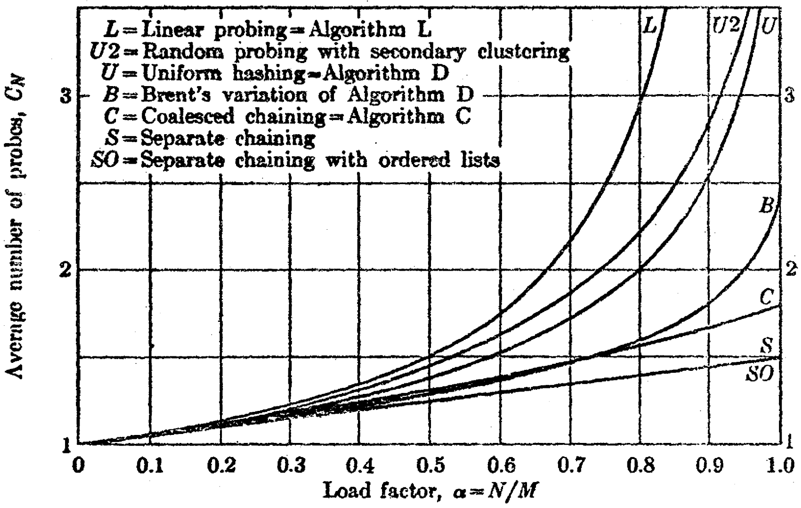
which is always worse than the performance of uniform probing, and much more so for load factors α near 1. The graphs in Figure 1.5.1., reproduced from [Knuth2], summarize these analyses. Algorithm S on these graphs refers to bucket search. We will have much more to say about the graphs for U2 and D in the following two chapters.

It is interesting to note that in all these hash methods, if we fix the load factor α, then the average number of probes is nearly independent of the table size m. Thus, loosely speaking, hashing allows us to retrieve records with a bounded number of probes, irrespective of the number of records we have. This is why it is so convenient to express the performance in terms of the load factor α, which is independent of m. It is also one of the reasons why hashing algorithms enjoy such popularity in software practice.

Figure 1.5.1. COMPARISON OF COLLISION RESOLUTION METHODS: LIMITING VALUES OF THE AVERAGE NUMBER OF PROBES AS M → ∞



(a) Unsuccessful search.



(b) Successful search

1.6. Summary of the Results

The purpose of this thesis is to investigate and illustrate techniques for the analysis of hashing algorithms. The traditional tools of algorithmic analysis are the tools of classical combinatorial enumeration: special numbers (i.e., binomial coefficients, Fibonacci numbers, etc.), recurrence relations, and generating functions. In Chapter 2 we apply these tools to a number of different hashing algorithms. In Section 2.1 we present an extension of Algorithm C that handles overflows with reasonable efficiency. In Section 2.2 we analyze a method combining open addressing and chained hashing; we prove that it is not as advantageous as it seems at first sight. Section 2.3 discusses k-ary clustering techniques for which C'_α is as large as possible and in fact turns out to grow linearly with m. The method used in the analysis indicates the power of the falling factorial power notation. The oracular argument by which the extremality of these techniques is proven is also of interest, as one of the few examples available where a certain algorithm in a class can be proven extremal *on the average.* We can see from the results of that section that only when k grows as $\Omega(\log m)$ will we be able to maintain C'_α bounded for the worst k-ary clustering techniques, as $m \rightarrow \infty$. On the other hand Section 2.4 shows that *on the whole* (i.e., *averaged* over *all* techniques) k-ary clustering techniques for k>1 are quite good. We prove that if the permutations described in the definition of k-ary clustering for k>1 are randomly chosen, then C'_α is asymptotically $1/(1-\alpha)$, the same as for uniform probing, which exhibits no clustering. We also analyze "random" secondary clustering (k=1), in which case we find that C'_α is asymptotically $1/(1-\alpha) - \alpha - \log(1-\alpha)$. Thus secondary clustering techniques on the average are worse than tertiary (since $\alpha + \log(1-\alpha) < 0$), although better than linear probing.

In Chapter 3 we exclusively concern ourselves with the analysis of double hashing. It has long been known from simulations that double hashing behaves essentailly identically with uniform probing. We prove that for $0<\alpha\leq\alpha_0$, where α_0 is an absolute constant, $\alpha_0 \sim .319$, this is indeed the case: C'_α for double hashing is $1/(1-\alpha) + O(1)$. This result is considerably deeper than any of the other analyses we have carried out and the techniques we use have a different flavor. We cannot appeal to recurrence relations or generating functions. Instead we use a probabilistic argument to prove that the configurations of αm occupied positions that double hasing gives rise to have almost always nearly the expected number of arithmetic progressions, and thus nearly the expected C'_α.

All of the results presented in Chapters 2 and 3 of this thesis are new, with the exception of the analysis of random secondary clustering that was previously done using a more complicated method in exercise 6.4-44 of [Knuth2]. Perhaps the most significant contribution is the method used in the analysis of double hashing, that allows one to prove the asymptotic equivalence in the performance between an algorithm and a simplified model of its behavior.

There are many open problems relating to the material of both chapters. Are there secondary clustering techniques with C'_α better than $1/(1-\alpha) - \alpha - \log(1-\alpha)$? If we let $h_2(K) = f(h_1(K))$ in double hashing, are there simple f (e.g., $f(x) = m-x$) for which the resulting algorithm is asymptotically equivalent to random secondary clustering? (Simulations support this contention for the f given in parentheses.) Are there *any* open addressing techniques with $C'_\alpha < 1/(1-\alpha)$ for almost all α? Most of these questions correspond to open problems in Section 6.4 of [Knuth2]. Perhaps we can prove that "almost all" k-ary clustering with $k>1$ techniques have a C'_α which is $(1\pm\varepsilon)/(1-\alpha)$. For Chapter 3 the most obvious next step is to extend the argument to work for all α, $0<\alpha<1$. The argument also can be applied to a modified double hashing algorithm, in which $h_2(K)$ is restricted to a linear segment of the table of size λm, for any fixed λ, $0<\lambda\leq 1$. The number of probes in the modified algorithm can be proven to be asymptotically equal to that of double hashing. This modified algorithm allows us to handle tables of non-prime size. Perhaps we can prove this for $h_2(K)$ restricted in any subset of size λm. A number of purely number theoretic questions about arithmetic in the finite field Z_m of m elements also arise (m is prime). We make the following two conjectures (1) Let l be fixed, $0<l<1/2$, $S = \{1,\ldots,m^l\}$, T any m^l-element subset of Z_m, $ST = \{st \mid s\in S, t\in T\}$. Then as $m \rightarrow \infty$, there exists a small constant ε such that $|ST|\geq m^{2l-\varepsilon}$; (2) if $0<x<m^{1/2}$, then no set of x elements of Z_m can have more than $O(x^2/k)$ arithmetic progessions of length k among its members, for any $k = 1,2,\ldots,x$. Settling either of these conjectures would prove double hashing equivalent to uniform hashing for all α.

The appendix at the end of the thesis illustrates a rather curious application of the exponential sums technique of analytic number theory to the problem of enumerating arithmetic progressions, which is treated by more combinatorial methods in Chapter 3.

"cogito ergo sum" = "I think, therefore I sum",
graffiti in CMU Science Hall restroom

CHAPTER 2:

RECURRENCE METHODS

In this chapter we present four different hashing analyses. These fall into the traditional paradigm of algorithmic analysis, that is they involve either direct counting or the recurrence relation/generating function techniques expounded in [Knuth1], [Knuth2]. In these algorithms we are fortunate to have probability distributions varying so regularly with the number of records in the table, that the desired performance either satisfies a closed form recurrence or can be explicitly calculated. The first two sections analyze two algorithms that allow one to handle overflows from a hash table gracefully. These methods work best if the number of records inserted is below a certain bound, but also handle larger quantities of records with only a moderate degradation in the performance. The last two sections discuss secondary, tertiary, and in general k-ary clustering techniques. Section 2.3 shows that the worst k-ary clustering techniques have an average number of probes linear in the table size. No fixed number of independent probes into the table can ever be sufficient by itself to guarantee sublinear performance. The subsequent probes *are* of paramount importance. In Section 2.4 the average performance over an entire class of hashing methods is computed. Perhaps most interesting is the result that a "random" k-ary clustering technique is asymptotically equivalent to uniform probing for $k>1$.

2.1. Chained Scatter Searching with Overflow.

A common way of resolving collisions in hashing is to chain, or link, together all records whose keys hash to the same address. When a record is to be looked up, the search begins at the table entry where its key hashes. If this entry is empty, then the record is not in the table and can be inserted right there. Each full entry in the table contains a link field which can be a pointer to another entry, or null, indicating the end of a chain. If the original entry in the table we came upon was full, and the record sitting there was unequal to our record, then this link field indicates the next entry to try. We keep repeating this process until either (1) we find the record we are looking for, or (2) we run off the end of the chain we are following. In case (2), we know the record is not in the table. In order to insert it, we first find an empty entry (or give up, if the table is full). Then we place the record there and plant a link to that entry in the last member of the chain we just exhausted. This completes the lookup process.

The above is an informal description of Algorithm C (*Chained scatter table search and insertion*) in Section 1.2. This algorithm is analyzed in exercises 6.4. 39-42 of [Knuth2]. The key results of the analysis are as follows: Let m be the total number of entries in the table and let n be the number of records currently in it. Assume that all m^n possible input hash sequences (i.e., sequences of possible hash values for the keys of our n records, in order of their insertion) are equally likely. Assume also that it is equally likely that we will search for any of the records presently in the table. Then the average number of comparisons in an unsuccessful search C'_n is given by

$$(1) \qquad C'_n = 1 + ((1 + 2/m)^n - 1 - 2n/m)/4 = 1 + (e^{2\alpha} - 1 - 2\alpha)/4 + O(1/m),$$

where $\alpha = n/m$ = occupancy factor of the table. For a successful search, the average number C_n of comparisons is

$$(2) \qquad C_n = 1 + (m/8n)\ ((1 + 2/m)^n - 1 - 2n/m) + (n-1)/4m$$

$$= 1 + (1/8\alpha)(e^{2\alpha} - 1 - 2\alpha) + \alpha/4 + O(1/m).$$

We proceed below to analyze a modification of Algorithm C that uses a table of size m' > m. Only the first m locations are used for hashing, so the first m'-m overflow items will go into the extra locations of the table. For fixed m', what choice of m in the range $1 \leq m < m'$ leads to the best performance?

We will need a number of facts from the answer to exercise 39 of Section 6.4 of [Knuth2]. A key quantity in the analysis of Algorithm C is the function $c(k_1,k_2,\ldots,k_n)$, which denotes the number of possible hash sequences of length n that give rise to k_1 chains of length 1, k_2 chains of length 2,...etc., in the table. Note that $1k_1 + 2k_2 + 3k_3 + \ldots = n$. These numbers satisfy a nice recurrence which can be used to show, among other things, that

$$\sum_{\substack{j\geq 1 \\ k_1+2k_2+\ldots = n}} k_j j^2 c(k_1,k_2,\ldots) = 1/2\ (m(m+2)^n - m^{n+1}). \tag{3}$$

To proceed, we observe that the new algorithm we propose, call it Algorithm C', behaves exactly like C, until the first m locations of the table are filled. Let the vector $(k_1,k_2,k_3,\ldots)$ represent the chain size distribution when this occurs (i.e., k_i = # of chains of length i, $k_1+2k_2+3k_3+\ldots = m$). Suppose that subsequently p overflow items have been inserted ($p\leq m'-m$). Each of these overflow entries will be part of a chain that originates in the regular hash area. Let the vector $(l_1,l_2,\ldots,l_m)$ denote the distribution of these items among the chains of various lengths, i.e., l_i = # of overflow items that have been "hung" from a chain that had length i when the regular part of the table was just filled up. Clearly, $l_1 + l_2 + l_3 + \ldots = p$. The following important observation states that the vectors $\underline{k}$ and $\underline{l}$ are adequate information for obtaining the average behavior of Algorithm C'.

Observation. As far as the average number of comparisons in an unsuccessful search is concerned, how the l_i overflow items (that got hung from chains of length i) are distributed among these k_i chains is immaterial.

Proof: We assume, of course, that an unsuccessful search is equally likely to originate in any of the m hash locations. Now if r overflow items get added to the tail of an i-chain, the effect is simply to increase the number of comparisons for an unsuccessful search by r, for each such search originating at one of the elements of the i-chain. Thus the total number of comparisons for all possible unsuccessful searches originating in that chain is incremented by ir. This is a linear function in r, and therefore, the total number of comparisons does not depend on the exact distribution of overflow items among chains of the same length. ▮

(Thus we can assume, without loss of generality, that all l_i overflow entries that have been attached to chains of length i have, in fact, been attached only to one of them.)

Now a chain with t regular elements and s overflow elements attached to its tail contributes to the total number of comparisons for *all* possible unsuccessful searches

$$(t+s) + (t-1+s) + \ldots + (1+s) = t(t+1)/2 + ts. \tag{4}$$

as illustrated in Figure 2.1.1.

As we noted, there are $c(k_1,k_2,\ldots)$ hash sequences of length m that give rise to a chain length distribution $\underline{k} = (k_1,k_2,\ldots)$. In how many ways can each such sequence be extended to one of length m+p, that gives rise to an overflow item distribution $\underline{l} = (l_1,l_2,\ldots)$?

Each of the l_i items that go to chains of length i must hash to one of the ik_i regular table locations occupied by these chains. This, together with the fact that the order in which the overflow items are inserted is unimportant (as far as $\underline{l}$ is concerned), shows that there are exactly

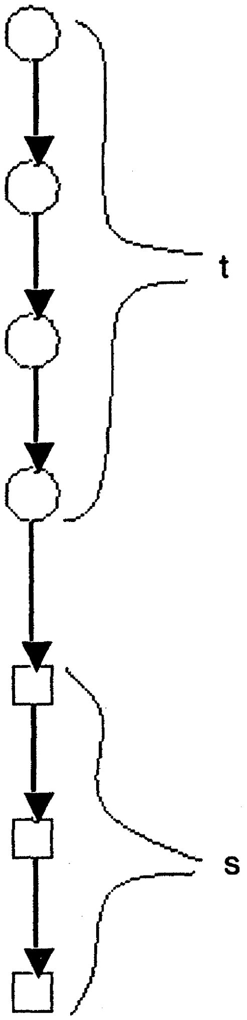
$$C(p; l_1,l_2,\ldots,l_m) \prod_{i=1}^{m} (ik_i)^{l_i}$$

ways to make the extension demanded above.

Using (4), we can now compute the total number of comparisons for all unsuccessful searches after p overflow items have been inserted, summed over all m^{m+p} possible input hash sequences. This number is

Figure 2.1.1.

A CHAIN WITH REGULAR AND OVERFLOW ELEMENTS



(5) $$\sum_{\substack{i\geq 1\\ l_1+l_2+\dots=p\\ k_1+2k_2+\dots=m}} ((k_i-1)i(i+1)/2 + \underbrace{i(i+1)/2 + il_i}_{\text{putting all } l_i \text{ overflow items on one of the } k_i\ i \text{ chains.}})\, C(p;\, l_1,l_2,\dots) \prod_{j=1}^{m} (jk_j)^{l_j}\, c(k_1,k_2,\dots)$$

We will use the following algebraic identity:

$$\sum_{i\geq 1} il_i\, C(p;\, l_1,l_2,\dots) \prod_{j=1}^{m} x_j^{l_j} = p(x_1+2x_2+\dots mx_m)(x_1+x_2+\dots+x_m)^{p-1}.$$

(Hint: $l_i C(p;\, \dots l_i \dots) = pC(p-1;\, \dots l_i-1 \dots)$).

So (5) can be rewritten as

$$\sum_{\substack{i\geq 1\\ k_1+2k_2+\dots=m}} \Big(\sum_{l_1+l_2+\dots=p} C(p;\, l_1,l_2,\dots) \prod_{j=1}^{m} (jk_j)^{l_j} \Big)\, k_i\, i(i+1)/2\, c(k_1,k_2,\dots) +$$

$$+ \sum_{k_1+2k_2+\dots=m} p(1^2k_1+2^2k_2+\dots+m^2k_m)\, m^{p-1}\, c(k_1,k_2,\dots) =$$

$$m^p \sum_{\substack{i\geq 1\\ k_1+2k_2+\dots=m}} k_i\, i(i+1)/2\, c(k_1,k_2,\dots) + pm^{p-1} \sum_{\substack{i\geq 1\\ k_1+2k_2+\dots=m}} k_i\, i^2\, c(k_1,k_2,\dots).$$

It is now not difficult to see that the left sum is simply the total number of comparisons for an unsuccessful search when the regular part of the table has been filled. Its value can be conveniently computed from (1) (let m = n and multiply by m^{m+1}). Similarly, the right sum is given to us by (3). To obtain the average number of comparisons for an unsuccessful search in Algorithm C', we need to divide the above expresion by m^{m+p+1} (namely m^{m+p} to account for all hash sequences, times m to account for all initial hash values of our particular search). Thus C'_p is

$$(6) \quad C'_p = (1 + 2/m)^m/4 + 1/4 + p(1 + 2/m)^m/2m - p/2m.$$

Note that when $p = 0$, this formula agrees with (1) if we set $n = m$, as it should. When $m = 1$, the above formula says that $C'_p = 1+p$ which is certainly true, for we then have a chain of $p+1$ records linked together, with every unsuccessful search necessarily starting at the head of the chain.

We can also observe that as $m \to \infty$, $(1 + 2/m)^m$ tends to e^2. In fact, $(1 + 2/m)^m$ increases monotonically to e^2, though rather slowly, the difference $e^2 - (1 + 2/m)^m$ being of the order m^{-1}. Thus for m sufficiently large the above formula can be approximated by

$$C'_p \approx (e^2 + 1)/4 + p(e^2 - 1)/2m \approx 2.097 + 3.194(p/m).$$

This formula gives us a rough idea of the degradation of search efficiency with increasing size of the overflow area. In a moment, we will have more to say on this point.

Let us also determine the average number of comparisons in a successful search by Algorithm C'. We will assume that each of the m+p table entries is equally likely to be sought. The key observation is that a successful search for a record uses one more comparison than the unsuccessful search made when the record was first inserted. There is one exception to this, namely when the record originally hashed to an empty location. Then we count *one* comparison for both the successful and unsuccessful searches. Thus the desired average is given by

$$C_p = (m + p - \sum_{k=0}^{m} (m-k)/k + \underset{\substack{\uparrow \\ \text{from (1)}}}{\sum_{k=0}^{m} C'_k} + \underset{\substack{\uparrow \\ \text{from (6)}}}{\sum_{k=m+1}^{m+p} C'_k}),$$

or

$$(7)\quad C_p = ((1 + m((1 + 2/m)^m/8 - 3) + m/4 - 1/4) +$$

$$+ p + ((1 + 2/m)^m - 1)p(p+1)/4m)/(m+p).$$

Note that when $p = 0$, this agrees with (2) if we set $n = m$. If $m = 1$, (7) gives $(p+2)/2$ which certainly is the average number of comparisons in a successful linear search down a list of length $p+1$.

If we use the fact that $p+m = m'$, we can rewrite (6) as

$$C'_p = (m'/2m - 1/4)\ ((1 + 2/m)^m - 1) + 1/2.$$

Keeping m' fixed, we can now ask for the m in the interval $1 \leq m \leq m'$ that minimizes C'_p. The approximation to e used above suggests that this occurs when $m = m'$. ($p/m = 0$, so $C'_p = 2.1$). It can be shown analytically that m' is a local minimum for m' sufficiently large: Briefly we have

$$(m'/2(m'-\Delta m) - 1/4)\ ((1 + 2/(m'-\Delta m)^{m'-\Delta m} - 1) + 1/2 =$$

$$(1/2 + \Delta m/2m' - 1/4)\ ((1+2/m')^{m'} - 2\Delta m/m'^2 - 1) + 1/2 =$$

$$C'_p + \underbrace{\Delta m(1+2m')^{m'}/2m' - \Delta m/m'^2 + \text{higher order terms.}}_{\text{This quantity positive for } m' \geq 1.}$$

This shows that for m' large, a small decrement in m ($= m'$) will increase C'_p.

A number of empirical tests were run on a computer for fifty values of m' between 10 and 1000. In all cases, the minimum occurrred for $m = m'$.

The above results establish that, from a time-efficiency viewpoint, there is nothing to be gained over Algorithm C by adding the overflow area.

Algorithm C', however, could still be useful in a situation where the number of entries to be put in the table only rarely exceeds a certain bound. In this situation, the storage allocated permanently to the table can equal the usual bound. Aditional storage need not be committed until really required, for Algorithm C' uses the overflow area only after the regular part of the table has been filled. And, as we saw in the above analysis, for any reasonable sized overflow area, the performance of our algorithm suffers only moderate degradation.

2.2. A Combined Open Address and Chained Hash Search.

In this section we offer an analysis of a hashing technique originally proposed by D. G. Bobrow. The algorithm is described in [Bobrow] as follows:

> "...the following nonhomogeneous algorithm seems to combine the best of both [open address and chained techniques]. In open address search, two keys are forbidden: 0 (say) which indicates an empty slot, and -1 (say) which indicates a deleted slot. In both these cases the value associated with the key is meaningless. If now instead of one area we allocate two, the first a one-shot (no collision) open address hash table, and the other a linearly allocated chain linked area, then we can use the first for open addressing. However, if there is a collision at an address then the pair at that address is moved to the linear collision (overflow) table. Then the key -1 is used to indicate that the value in the open address table corresponding to this key is a pointer into the overflow area.
>
> Thus when the table is mostly empty it acts as an open address scheme, and when it is almost full it acts like a bucket search. Ordering can be used on the overflow lists to cut down search times for failure. The collision case obviously takes exactly one more probe than it would in the chain hash search [to discover it is the chain case] ..."

Bobrow's technique uses a fixed hash table and an overflow area that is allocated dynamically. In its simplest form the technique maintains the overflow area as a set of disjoint linked lists, each list consisting of keys that have hashed to the same address of the fixed table. In case of a collision both colliding entries are moved to the overflow area and a special key (say -1) is stored at the h(K)-th entry, indicating that the value field of that entry is a pointer to the beginning of a list in the overflow area. Thus records are stored in the fixed table until a collision occurs at their position, in which case they are moved to the overflow. For convenience of description we shall term this method Algorithm B in what follows.

In the analysis of Algorithm B the following quantities are important:

m = size of fixed table (number of entries),

n = number of records inserted so far,

$\alpha = n/m$ = the load factor of the table

k = size of the key field of an entry (say in words),

v = size of the value field of an entry (clearly if v is big it is best to store only a pointer to it in the table itself), and

l = size of the link field of an entry.

Algorithm B assumes that $v \geq l$ and uses $m(k+v)$ words of fixed table and $k+v+l$ words per entry in the overflow area. One of the desirable properties of this algorithm is that no link field need be allocated in the fixed table; thus if the overflow area is only rarely used, Algorithm B offers us the storage economy of an open address technique, together with the speedy search of a chained method. We will use the quantity

$$\beta = l/(k+v+l)$$

to represent the average savings factor of our technique.

In the analyses below we assume that all m^n possible sequences of the hash values of the n inserted keys are equally likely.

Perharps the most interesting of the computations is that of the average size of the overflow area in Algorithm B. The number of entries in the overflow area equals the number of keys inserted at whose hash address at least one collision has occurred. Let p(m,n,k) represent the probability that exactly k non-special (i.e., $\neq 0,-1$) keys are present in the fixed table. In this event the overflow area has size n-k.

A given sequence of n keys to be inserted can be thought of as a mapping from a set of n objects (the keys) to a set of m objects (the possible hash addresses). Any such mapping defines a partition of its domain into disjoint classes, each class containing all keys hashing to the same address. In this model we can interpret p(m,n,k) as the probability that a random mapping has exactly k classes of size 1. Let $Q(m,n,k) = m^n p(m,n,k)$ = number of mappings with the given property. Then, by separating out the k elements that form the singleton classes, we obtain the recurrence

$$(1) \qquad Q(m,n,k) = C(m,k)\ C(n,k)\ k!\ Q(m-k,n-k,0)$$

The numbers Q(m,n,0) count the number of mappings with no singleton classes. By selecting one of the elements in the range of such a mapping we can get the recurrence

$$(2) \qquad Q(m,n,0) = \sum_{0\le t\le n} C(n,t)\ Q(m-1,t,0)\ (1 - \delta_{(n-t)1}).$$

Since we cannot explicitly solve (1) and (2) we now proceed to introduce the exponential generating functions [Liu]

$$G_m(z) = \sum_{n\ge 0} C(m,n,0)\ z^n/n! \qquad \text{and}$$

$$G_{mk}(z) = \sum_{n\ge 0} C(m,n,k)\ z^n/n!\ .$$

We have $G_1(z) = e^z - z$, because when m = 1 there exists exactly one mapping for all n, except when n = 1, in which case our condition is violated.

Equation (2) translates to the recurrence for G_m

$$G_m(z) = G_{m-1}(z)\ (e^z - z) \quad , \qquad m>1$$

which coupled with the above remark implies that

$$G_m(z) = (e^z - z)^m.$$

From this and (1) we finally get the generating function we want

$$G_{mk}(z) = C(m,k)\ z^k\ (e^z - z)^{m-k}.$$

The average number of non-special keys present in the fixed table is

$$A_n = \sum_{0 \leq k \leq n} k\ p(m,n,k) = 1/m^n \sum_{k \geq 0} k\ Q(m,n,k).$$

By interchanging the order of summation we obtain

$$\sum_{n>0} m^n A_n z^n / n! = \sum_{k>0} k\ C(m,k)\ z^k\ (e^z - z)^{m-k} = mze^{(m-1)z}.$$

Therefore $A_n = n(1 - 1/m)^{n-1}$, and so the average size of the overflow area is

$$n(1 - (1 - 1/m))^{n-1}.$$

By similar manipulations we can compute the variance of the size of the overflow area to be

$$V_n = \sum_{0 \leq k \leq n} (A_n - k)^2\ p(m,n,k) =$$

$$= n(n-1)(1 - 1/m)(1 - 2/m)^{n-2} + n(1 - 1/m)^{n-1} - n^2(1 - 1/m)^{2n-2},$$

which is rather small (the n^2 terms cancel).

The computation of C_n and C'_n, the average number of comparisons done in a successful or unsuccessful search respectively, proceeds along standard lines.

Let A_t = the average number of comparisons needed to find the t-th key that was inserted. If $p_t(r)$ denotes the probability that the list to which the t-th key hashes has r-1 entries in the overflow area, then

$$p_t(r) = C(t-1,r-1)\ (1/m)^{r-1}\ ((m-1)/m)^{t-r} \text{ and}$$

$$A_t = \sum_{r\geq 1} (r+1)p_t(r) - p_t(1) = 2 + (t-1)/m - ((m-1)/m)^{t-1}.$$

Thus

$$C_n = 1/n \sum_{1\leq t\leq n} A_t = 2 + (n-1)/2m - (m/n)(1 - (1 - 1/m)^n).$$

Similarly for an unsuccessful search, let $p'_n(k)$ = probability that the list we search has k entries in the overflow area. Then

$$p'_n(k) = C(n,k)\ (1/m)^k\ ((m-1)/m)^{n-k} \text{ and}$$

$$C'_n = \sum_{k\geq 0} (k+1)p'_n(k) - p'_n(1) =$$

$$= 1 + (n/m)(1 - (1 -1/m)^{n-1}).$$

In summary, the asymptotic performance of Algorithm B is given by

average number of comparisons
for a successful search $= 2 + \alpha/2 - (1 - e^{-\alpha})/\alpha$,

and

average number of comparisons
for an unsuccessful search $= 1 + \alpha(1 - e^{-\alpha})$,

where the given values are in fact approximations with an error of the order of $O(1/m)$.

Our storage analysis has proved that

average number of entries

in the overflow area of
Algorithm B = $n(1 - (1 - (1/m))^{n-1}) = n(1 - e^{-\alpha}) + O(1)$,

and the distribution has a rather small variance, i.e. the the variance equals $(e^{-\alpha}+\alpha)n + O(1)$.

In order to interpret these numbers we compare Algorithm B with two other methods, a pure bucket search, in which the fixed table consists only of list-heads, and Algorithm C. of Section 1.2 in which there is no overflow area and the lists are maintained in the fixed table itself. The bucket search uses ml locations for the listheads and $n(k+v+l)$ for the entries, whereas Algorithm C uses just a fixed table of size $m(k+v+l)$.

The two graphs of Figure 2.2.1. compare the average time spent in a search by each of these hash methods. Under the reasonable assumption that look-ups will be a lot more frequent than insertions, we conclude that the time performance of Algorithm B is relatively disappointing. It appears that at every load factor a successful search by Algorithm B wil be more expensive on the average than one by the other two methods. This is so because an extra comparison with the special key (-1) will be involved in all successful searches except those succeeding in the first probe. The presence of this special key will also complicate the programming of the search loops in Algorithm B.

The table below indicates, for various savings factors β, the load factor α at which the average storage requirements of Algorithm B exceed those of Algorithm C. (It can be shown that for $m \geq 500$ these load factors α vary with m only within a tolerance of 10^{-3}).

β	α
.1	.344
.3	.637
.5	.864
.7	1.00
.9	1.00

Figure 2.2.2. plots the storage requirements of these three techniques in the case $k = v = l$ (so $\beta = 1/3$). (Table size m = 1000 was used, but the relative magnitudes of the graphs are fairly independent of m.) The storage requirements of Algorithm B are sandwiched between those of Algorithm A and Algorithm C, with a cross-over point at approximately $\alpha = .67$. Thus we can conclude that in the case $k = v = l$ Bucket Search has a better running time and simpler progrm than Algorithm B, at the cost of only slightly worse storage economy at high load factors. Therefore Bucket Search is preferable, unless very special information

"." REPRESENTS ALGORITHM A
"*" REPRESENTS ALGORITHM B
"#" REPRESENTS ALGORITHM C

```
  8[                                                              .
   [                                                           ..**
  6[                                                       ..****
   [                                                    ..***
  4[                                               ...****
   [                                             .****
  2[#########################################.****#################
   [                                     *****
  0[                                 ****.
   [                           ******.
  8[                     ******....
   [             ********   ...
  6[*************       ....
   [                ....
  4[             ...
   [         ....
  2[      ...
   [  ....
  0[..
   -------------------------------------------------------------------
    0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4
XORG=1.0 YORG=1003.0 XDELTA=16.0 YDELTA=170.0
XMAX=993.0 YMAX=3979.0
```

AVERAGE AMOUNT OF STORAGE USED AS A FUNCTION OF THE NUMBER N OF ENTRIES IN THE TABLE; HERE M = 1000

Figure 2.2.1. COMPARISON OF THREE ALGORITHMS

```
  8[                                                              #
   [                                                             #
  6[                                                            #
   [                                                          ##
  4[                                                         #
   [                                                       ##
  2[                                                      #
   [                                                    ##
  0[                                                  ##       ****
   [                                                ##     ****
  8[                                              ##   ****
   [                                           ### ****
  6[                                         ##****            ....
   [                                     #*****          ......
  4[                                 *****        .......
   [                           ******#    ........
  2[                     ******##.........
   [            *********........
  0[************....
   -------------------------------------------------------------------
    0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4
XORG=0.0 YORG=1.0 XDELTA=0.016 YDELTA=0.06
XMAX=0.99200006 YMAX=2.0719432
```

AVERAGE NUMBER OF COMPARISONS FOR AN UNSUCCESSFUL SEARCH AS A FUNCTION OF THE LOAD FACTOR α

Figure 2.2.1. (CONTINUED)

```
  8[*
   [
  6[
   [                                                          *****
  4[                                                      ****   ##
   [                                                  ****    ###
  2[                                             *****     ###
   [                                         ****       ###
  0[                                     ****       ####
   [                                 ****        ###            ...
  8[                             ****        ####        .......
   [                         ****        ####     .......
  6[                     ****       #####  .......
   [                 ****      #####.......
  4[             ****     ###.......
   [         ****   ##.......
  2[      *** #.......
   [  ****.....
  0[.*..
   -------------------------------------------------------------------
    0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4
XORG=1.0E-9 YORG=1.0 XDELTA=0.016 YDELTA=0.056
XMAX=0.99200006 YMAX=2.0
```

AVERAGE NUMBER OF COMPARISONS FOR A SUCCESSFUL SEARCH AS A FUNCTION OF THE LOAD FACTOR α

Figure 2.2.1. (CONTINUED)

concerning the keys and the searches is given which allows us to conclude that for this particular application Algorithm B is more desirable.

2.3. The Worst k-ary Clustering Techniques

In this section we will discuss k-ary clustering techniques whose average number of probes grows *linearly* with the table size. One obvious way to cause substantial clustering is to force the probe paths of all keys to be identical after the first k random probes have failed. The technique described below is a natural generalization of that presented in exercise 6.4-46 of [Knuth2].

The hashing technique uses k distinct independent random probes, first at $h_1(K)$, the second at $h_2(K)$, where $h_2(K) \neq h_1(K)$, etc., up to a probe at $h_k(K)$. If we have collisions in all these probes, then we make a left to right scan of the table for an empty slot, skipping over any positions already examined.

As usual we let m = table size and n = number of entries in the table. The key quantity is

$H(m,n,j)$ = number of possible hash sequences of length n
[more precisely, sequences of n k-tuples $(h_1(K),h_2(K),\ldots h_k(K))$]
such that locations $1,2,\ldots,j$ of the table get occupied and location $j+1$ is empty.

By considering the remaining n-j entries we obtain

$$H(m,n,j) = C(m-j-1,n-j)\ F(m,n,j),$$

where $F(m,n,j)$ denotes the number of sequences of n k-tuples $(h_1(K),h_2(K),\ldots h_k(K))$ that occupy positions $0,1,\ldots,j-1,j+1,\ldots,n$ (thus leaving position j empty). We observe that the numbers $F(m,n,j)$ satisfy the following recurrence based on n:

$$(*) \qquad F(m,n+1,j) = (f(m,n) + n^{\underline{k}}) \sum_{0 \le l < j} F(m,n,l) + (n+1-j)f(m,n)F(m,n,j),$$

where $f(m,n) = (m^{\underline{k}}-n^{\underline{k}})/(m-n) = n^{\underline{0}}(m-1)^{\underline{k-1}} + n^{\underline{1}}(m-2)^{\underline{k-2}} + \dots + n^{\underline{k-1}}(m-k)^{\underline{0}}$.

(This last identity is left as an exercise to the reader).

To see this we distinguish two cases:

Case 1: When n keys are present, position j is empty and all the other positions 0,1,...,j-1 occupied. Now note that F also counts the number of hash sequences that occupy positions 0,1,...,j-1 leave position j empty, and then occupy *any fixed n-j positions* among j+1,...,m-1. Before the (n+1)-st key is inserted, we must have a situation in which positions 0,1,...,j-1 are occupied, position j is empty, and among positions j+1,...,n+1 exactly one is empty. This empty position has to be filled with the next insertion. As there are n-j+1 such candidate positions, this can happen in $(n-j+1)(m-1)^{\underline{k-1}}$ ways on the first probe, or $(n-j+1)n(m-2)^{\underline{k-2}}$ ways on the second probe, ..., or $(n-j+1)n^{\underline{k-1}}$ on the last of the random probes. This covers all possible ways, and so we get the term $(n+1-j)f(m,n)F(m,n,j)$.

Case 2: When n keys are present, position *l* is empty, *l* < j, positions 0,1,...,*l*-1 are occupied, position j is empty, and positions *l*+1,...,j-1,j+1,...,n+1 are occupied. The (n+1)-st insertion has to fill position *l*. This can happen on the first probe in $(m-1)^{\underline{k-1}}$ ways, on the second probe in $n(m-2)^{\underline{k-2}}$ ways, ..., on the k-th probe in $n^{\underline{k-1}}$ ways, and finally on succeeding probes (i.e. during the linear scan from left to right) in $n^{\underline{k}}$ ways, for a total of $f(m,n)+n^{\underline{k}}$ ways. Summing this contribution over all *l* we obtain the first term on the right hand side of (*).

The proper initial values for this recurrence are easily seen to be

$$F(m,1,0) = F(m,1,1) = (m-1)^{\underline{k-1}}.$$

If C'_n denotes the average number of comparisons to insert a new element in the table when n are already present, then clearly

$$C'_n = 1 + \sum_{0 \le j \le n} H(m,n,j)P(m,n,j)/(m^{\underline{k}})^n,$$

where

$P(m,n,j)$ = average number of collisions when inserting a random k-tuple $(h_1(K),h_2(K),\dots h_k(K))$ in a configuration of n occupied positions of the type counted by $H(m,n,j)$.

The quantity P(m,n,j) can be broken down according to the number of collisions that occured during the random probes:

$$P(m,n,j) = (m-n) \sum_{0 \le i < k} i n^{\underline{i}}/m^{\underline{i+1}} + (n^{\underline{k}}/m^{\underline{k}})(j + k(n-j)/n);$$

here j + k(n-j)/n is the average number of collisions *after* all k random probes have failed.

This sum can be computed in closed form by expressing the falling powers in terms of factorials and then in terms of binomial coefficients and finally appealing to the well-known identity 1.2.6-11 of [Knuth1]. Since the manipulations are similar to those illustrated below for the more important sum of the H's, they will be omitted. The result is

$$P(m,n,j) = \frac{nm^{\underline{k}} - (n + k(m-n))n^{\underline{k}}}{m^{\underline{k}}(m-n-1)} + (n^{\underline{k}}/m^{\underline{k}})(j + k(n-j)/n).$$

Therefore

$$C'_n = 1 + \frac{nm^{\underline{k}} - (n + k(m-n))n^{\underline{k}}}{m^{\underline{k}}(m-n-1)} + kn^{\underline{k}}/m^{\underline{k}} + $$
$$+ ((n-1)^{\underline{k}}/(m^{\underline{k}})^{n+1}) \sum_{0 \le j \le n} jH(m,n,j).$$

We have used above the fact that

$$\sum_{0 \le j \le n} H(m,n,j) = (m^{\underline{k}})^n,$$

but it will be useful for our purposes to verify this directly. We will use the recurrence relation (*) to prove this fact inductively.

We have for n=1

$$\sum_{0\leq j\leq 1} H(m1,j) = H(m,1,0) + H(m,1,1) =$$

$$(m-1)F(m,1,0) + F(m,1,1) = m.(m-1)^{\underline{k-1}} = m^{\underline{k}},$$

as desired.

Now

$$\sum_{j} H(m,n+1,j) = \sum_{j} C(m-j-1,n+1-j)F(m,n+1,j) =$$

$$\sum_{j} \sum_{0\leq l<j} C(m-1-j,n+1-j)(f(m,n)+n^{\underline{k}})F(m,n,l) +$$

$$\sum_{j} C(m-1-j,n-j)(m-n-1)f(m,n)F(m,n,j) =$$

$$\sum_{0\leq l<m} F(m,n,l)[(f(m,n)+n^{\underline{k}})\sum_{l<j<m} C(m-1-j,n+1-j) + (m-n-1)f(m,n)C(m-1-l,n-l)] =$$

$$\sum_{0\leq l<n} F(m,n,l)C(m-1-l,n-l)[f(m,n) + n^{\underline{k}} + (m-n-1)f(m,n)] =$$

$$m^{\underline{k}} \sum_{0\leq l<n} F(m,n,l)C(m-1-l,n-l) = m^{\underline{k}} \sum_{j} H(m,n,j).$$

This completes the inductive proof.

Next we evaluate $\Sigma_{0\leq j\leq n}$ jH(m,n,j) in a similar manner.
We have

$$\sum_{j} jH(m,n+1,j) = \sum_{j} jC(m-j-1,n+1-j)F(m,n+1,j) =$$

$$\sum_{j} \sum_{0 \le l < j} C(m-1-j,n+1-j)(f(m,n)+n^{\underline{k}})jF(m,n,l) + \sum_{j} C(m-1-j,n-j)j(m-n-1)f(m,n)F(m,n,j) =$$

$$\sum_{0 \le l < n} F(m,n,l)[(f(m,n)+n^{\underline{k}})\sum_{l < j < m} jC(m-1-j,n+1-j) + (m-n-1)f(m,n)C(m-1-l,n-l)l] =$$

$$\sum_{0 \le l < n} F(m,n,l)[C(m-1-l,n-l)(f(m,n)+n^{\underline{k}})(m+(m-n-1)k))/(m-n) + (m-n-1)f(m,n)C(m-1-l,n-l)l] =$$

$$[m^{\underline{k}} - (f(m,n)+n^{\underline{k}})/(m-n)] \sum_{0 \le l < n} lH(m,n,l) + m(m^{\underline{k}})^{n}(f(m,n)+n^{\underline{k}})/(m-n).$$

Thus if we write

$$A_n = \sum_{j} jH(m,n,j)/(m^{\underline{k}})n,$$

then we have shown that

$$A_{n+1} = [1 - (f(m,n)+n^{\underline{k}})/m^{\underline{k}}(m-n)]\ A_n + m(f(m,n)+n^{\underline{k}})/m^{\underline{k}}(m-n).$$

It is easy to show that $A_1 = 1/m$.

It is perhaps best to write this recurrence as

$$A_{n+1}/m = [1 - (f(m,n)+n^{\underline{k}})/m^{\underline{k}}(m-n)]\ A_n/m + (f(m,n)+n^{\underline{k}})/m^{\underline{k}}(m-n),$$

from which it is easy to verify that

$$A_n = m - (m-1/m) \prod_{i=1}^{n-1} [1 - (f(m,i)+i^{\underline{k}})/m^{\underline{k}}(m-i)]$$

$$= m - (1+1/m)(m-n) \prod_{i=1}^{n-1} [1 + (1-i^{\underline{k}}/m^{\underline{k}})/(m-i)].$$

(We interpret the empty product to be 1).

Thus we finally have an explicit formula for C'_n, namely,

$$C'_n = 1 + \frac{nm^{\underline{k}} - (n + k(m-n))n^{\underline{k}}}{m^{\underline{k}}(m-n-1)} + kn^{\underline{k}}/m^{\underline{k}} + ((n-1)^{\underline{k}}/m^{\underline{k}})[m - (1+1/m)(m-n) \prod_{i=1}^{n-1} [1 + (1-i^{\underline{k}}/m^{\underline{k}})/(m-i)]].$$

Since this is a rather complicated expression, we proceed now to derive the asymptotic growth of C'_n for $n = \alpha n$, α fixed, $m \to \infty$. Let us look at the asymptotics of the product. Taking logarithms we have, since $f(m,i)/m^{\underline{k}} = O(1/m)$,

$$- \sum_{1 \leq i < n} \log(1+(1-i^{\underline{k}}/m^{\underline{K}})/(m-i)) = - \sum_{1 \leq i < n} f(m,i)/m^{\underline{k}} + O(1/m) =$$

$$- \sum_{1 \leq i < n} \sum_{0 \leq j < k} i^{j}/m^{\underline{j+1}} + O(1/m) = - \sum_{0 \leq j < k} \sum_{1 \leq i < n} i^{j}/m^{\underline{j+1}} + O(1/m) =$$

$$- \sum_{0 \leq j < k} n^{\underline{j+1}}/(j+1)m^{\underline{j+1}} + O(1/m) =$$

$$- \sum_{1 \leq j \leq k} \alpha^{j}/j + O(1/m).$$

Substituting into the formula for C'_n we see that

$$C'_n = \alpha^{k}(1-(1-\alpha)\exp(-\Sigma_{1 \leq j \leq k} \alpha^{j}/j))\ m + O(1), \text{ as } m \to \infty.$$

Thus we can see that C'_α increases linearly with the table size m. The implied constant above can be made absolute if we restrict α to any interval $0 \leq \alpha \leq \alpha_0$, for any absolute constant $\alpha_0 < 1$. Thus we have

THEOREM 2.3.1. A k-ary clustering technique that probes the table in a fixed sequence after the first k random probes have been exhausted (naturally ignoring any positions in this sequence already examined) has a performance that as $m \to \infty$, $n = \alpha m$, $0 < \alpha < 1$, and α, k fixed given by

$$C'_n = \qquad \alpha^k(1-(1-\alpha)\exp(-\Sigma_{1 \leq j \leq k}\ \alpha^j/j))\ m + O(1).$$

Proof. Clearly the above argument applies for any fixed permutation of probing the table entries after the intitial k probes. ▮

From the above analysis we see that k has to get as high as $\Omega(\log m)$ before we can hope to cause these worst k-ary clustering techniques to have a bounded number of comparisons on the average, as $m \to \infty$.

We now prove that the above considered techniques are indeed the worst possible k-ary clustering techniques, that is they have the largest possible *average* number of probes per insertion. The oracular argument given below is interesting, as it is one of the few examples where we can prove certain algorithms extremal on the average. To prove this we will find it easier to prove a somewhat stronger result.

THEOREM 2.3.2. For all fixed n, among all hashing techniques that begin the search into the table with k independent random probes and then continue the search in any manner whatsoever, no methods will exhibit a worse C'_n than those which after the initial k probes follow a fixed permutation, skipping over any positions already examined.

Proof. By just renaming the table positions it is immediately clear that the performance of the the method is independent of which specific permutation we use, and thus the "worst performance" mentioned above is well defined.

For the proof we shall think of a method under consideration as being defined by an oracle which, when the initial k random probes occur, decides, on the basis of the size of the table, the key being inserted, and even perhaps by consulting random variables, which of the remaining empty positions to fill.

Let us number the table entries 0,1,2,...,m-1 according to the sequence in which they are encountered during the final test search, i.e. during the insertion of the (n+1)-st key. Thus without loss of generality in the sequel we will be thinking of the final search as being a linear scan of the table from left to right. The number of comparisons made will then be the length of the leftmost contiguous group of occupied positions.

The assertion of the theorem is equivalent to stating that the oracle $\underline{O}$ that always fills the leftmost available slot in the table causes the average length of the leftmost group of occupied positions to be at least as large as any other oracle.

In each case the table gets filled by a sequence of moves which are of two kinds: either an R-move (random) in which an empty position is filled during the initial k random probes, or an O-move (oracle) in which an empty slot is filled by the oracle. Note that

> (1) At each moment each empty slot is equally likely to be filled by an R-move
>
> (2) At each insertion the probablility that it wil be an R- vs an O-insertion is independent of the oracle we are using.

We will now represent the sequence of insertions by a string over the integers among 1,...,m and the character "O". The integer i will indicate an R-move that fills the i-th available position from the left. The "O" will indicate an O-move. For example, with a certain oracle, the following insertion sequence S

$$5 \quad 6 \quad O_1 \quad 3 \quad O_2 \quad 2 \quad 1 \quad 2$$

might fill the table as follows (the subcripts of the R's are the corresponding R-moves; those of the O's are just used to distinguish the different oracle moves):

$$|R_1|R_2|R_3|O_1|R_5| \quad |R_6|O_2|R_2| \ldots$$

whereas our "worst" oracle $\underline{O}$ would produce the configuration

$$|O_1|O_2|R_1|R_3|R_5|R_2|R_6| \quad |R_2| \;\dots\; .$$

The above remarks imply that, given any two oracles O^1 and O^2 the above way of representing insertions is a 1:1 correspondence of the ways of filling the table with the two oracles such that the corresponding ways are *equiprobable*.

The following inductive assertion will prove that our oracle $\underline{O}$ leads to an initial contiguous occupied group that is at least as long as that produced by any other oracle O *for each insertion specification sequence S*, and thus prove the Theorem.

Assertion: For each n, for each insertion sequence S, and for each i, $1 \leq i \leq m-n+1$, when the n-th insertion is made, the i-th empty position from the left will not be a lower numbered entry of the table if we were using our oracle $\underline{O}$ as compared with any other oracle O.

The above example illustrates this principle. For the proof we induct on n. We can trivially check it for n=1. So all that remains is to verify that our assertion is preserved when the n-th key is inserted. Now if this new insertion is an R-move the effect is to simply take out a pair of corresponding positions in the two configurations, so the assertion clearly remains true. See Figure 2.3.1., part (a).

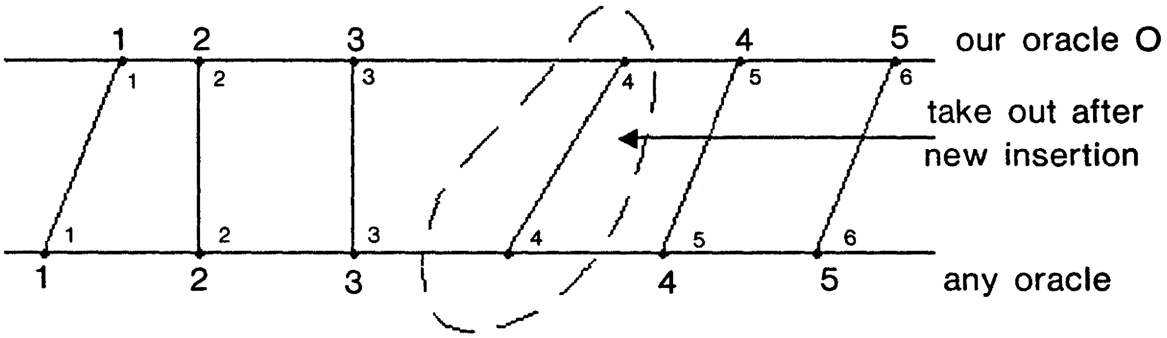
If the new insertion is an O-move, then again Figure 2.3.1., part (b) makes it obvious why the assertion is preserved.

Thus our proof of the theorem is complete, by applying the assertion for i=1. ∎

2.4 The Analysis of Random k-ary Clustering Techniques

In this section we will determine the performance of a k-ary clustering technique, when the permutations describing the method are chosen at random. What we mean by this is that we consider all possible $(m-1)!^m$ selections of m permutations defining a secondary clustering matrix such as described in Section 1.4. as equally likely. Similarly, we have $(m-2)!^{m(m-1)}$

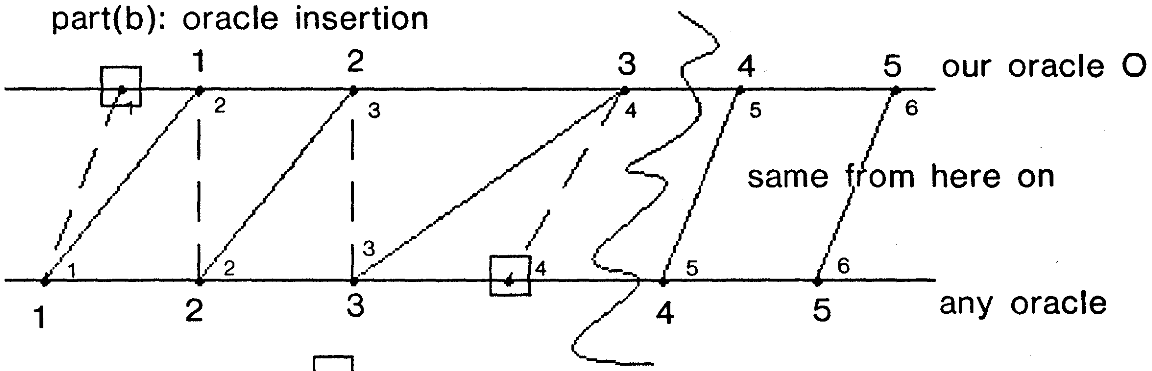
part (a): empty positions numbered from left to right
random insertion


1 2 3 4 5
our oracle O
1 2 3 4 5 6
take out after
new insertion
1 2 3 4 5 6
1 2 3 4 5
any oracle

1,2,... = new numbering

1,2,... = old numbering

part(b): oracle insertion


1 2 3 4 5
our oracle O
1 2 3 4 5 6
same from here on
1 2 3 4 5 6
1 2 3 4 5
any oracle

□ = position filled

Figure 2.3.1. THE ORACLE ARGUMENT

choices in the case of tertiary clustering, and so on. The analysis of random secondary clustering was first carried out correctly in exercise 6.4-44 of [Knuth2] by a considerably more complicated method. The results we get are interesting because they provide us with an idea of how other methods very similar to linear probing and double hashing might perform. We find, for example, that the random k-ary clustering technique with k>1 has

$$C'_n = 1/(1-\alpha) + O(1/m), \quad n = \alpha m,\ m \to \infty.$$

This is interesting from several viewpoints: 1) as we saw in Section 2.3., there are k-ary clustering techniques whose performance is very bad, that is for which C'_n grows linearly with m; 2) as we remarked earlier, $1/(1-\alpha)$ is also C'_n for uniform hashing, which implies, for example, that *on the average* tertiary clustering techniques are very good, since they perform just as well as uniform hashing which needs to compute an arbitrary number of independent hash functions, not just two. For secondary clustering we find that

$$C'_n = 1/(1-\alpha) - \alpha - \log(1-\alpha) + O(1/m),$$

which is significantly better than linear probing. This can be realized in practice by letting $h_2(K) = f(h_1(K))$ in the double hashing algorithm, where f is some easily computable function.

We now begin the mathematical portion of this section where we perform the analyses described. The calculations are somewhat involved, so we will emphasize only the key steps. Let $\rho_{n,s}$ denote the probability that when the (n+1)-st key is inserted in the table s probes will be made. We let $p = 1/m^{\underline{k}}$, $q = 1-p$, and make use of our falling-factorial power calculus. The following two identities are easily proved by induction or by techniques analogous to those used in Section 2.3.:

$$\text{(I1)} \qquad \sum_{0\le t\le a} a^{\underline{t}}/b^{\underline{t+1}} = 1/(b-a), \quad \text{and}$$

$$\text{(I2)} \qquad \sum_{0\le t\le a} k a^{\underline{t}}/b^{\underline{t+1}} = a/(b-a+1)(b-a),$$

where we interpret $x^{\underline{0}} = 1$. Let us now look at Figure 2.4.1. which illustrates the matrix defining a secondary clustering technique. The basic observation is that we need not know

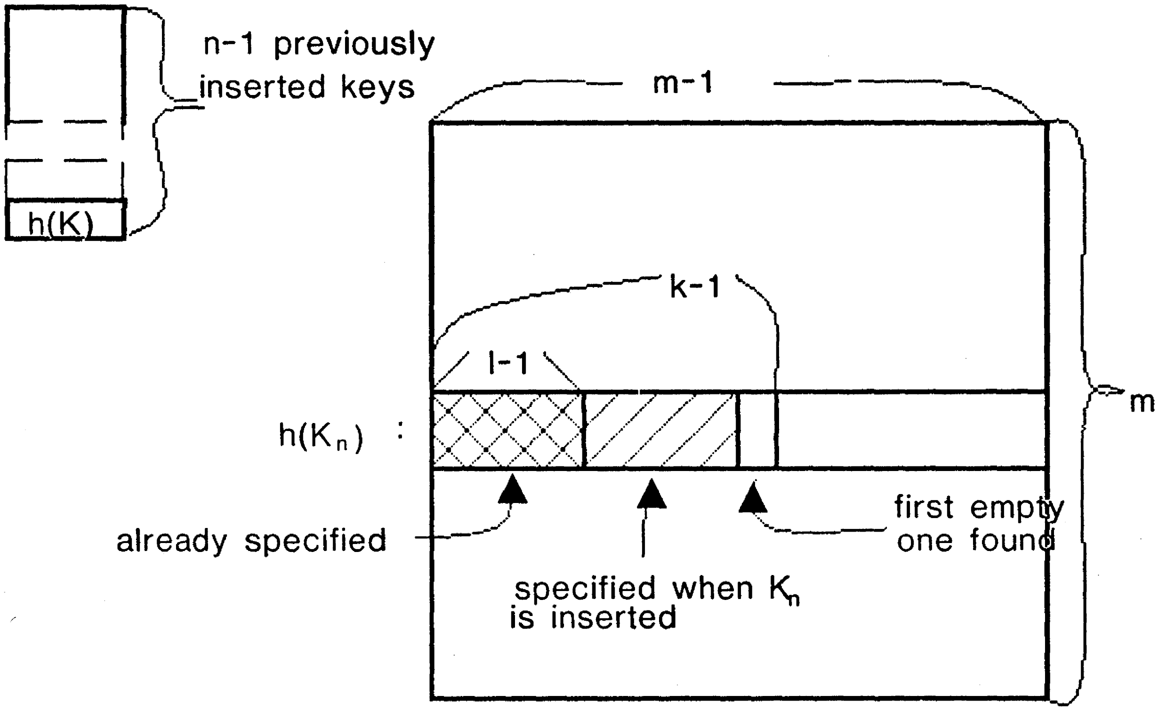

n-1 previously
inserted keys
h(K)
m-1
m
k-1
l-1
h(Kn) :
already specified
specified when Kn
is inserted
first empty
one found

Figure 2.4.1.
THE MATRIX DEFINING A
SECONDARY CLUSTERING TECHNIQUE

what an element of this matrix is, until we have a need to look at it. When the (n+1)-st key K_{n+1} is inserted, the only row of the matrix that is of interest to us is row $h(K_{n+1})$, in which case a portion of this row has already been specified. Specifically assume that t-1 initial elements of this row have been specified. Since we want to compute the probability of making s probes, it follows that we will have to specify s-t additional elements of this row. The first s-t-1 must specify table positions that have already been occupied, whereas the last one must specify an empty position. We now break $\rho_{n,s}$ into a number of components, according to the greatest $i \leq n$ for which $h(K_i) = h(K_{n+1})$. We further note that the probability of filling the remaining s-l positions of the row as described above when s>k (in this case k=1) is

$$(m-n)(n-t)^{\underline{s-t-1}}/(m-t)^{\underline{s-t}} \ ,$$

because for the t-th we have n-t choices of occupied positions to be selected among the m-t possibilities for continuing this row, for the (t+1)-st we have n-t-1 choices among the m-t-1, and so on till the last one. This last one, however, is exceptional because it must hit an empty position, so we have m-n choices out of m-s+1. As a result of all this we can write down a recurrence relation for the probabilities $\rho_{n,s}$. For the general k-ary clustering case we obtain the following:

$$\rho_{n,1} = (m-n)/m,$$

$$\rho_{n,2} = n(m-n)/m(m-1),$$

......

$\rho_{n,k} = (m-n)n^{\underline{k-1}}/m^{\underline{k}}$, since the first k probes are independent and random, and for s>k we have

$$\rho_{n,s} = (m-n)\Big[\sum_{1 \leq i \leq n} pq^{n-i} \sum_{k \leq t < s} \rho_{i,t}\, (n-t)^{\underline{s-t-1}}/(m-t)^{\underline{s-t}} +$$

$$+\Big(1 - \sum_{1 \leq j \leq k} \rho_{n,j} - \sum_{1 \leq i \leq n} pq^{n-i}\Big(1 - \sum_{1 \leq j < k} \rho_{i,j}\Big)\Big)(n-k)^{\underline{s-k-1}}/(m-k)^{\underline{s-k}} \Big].$$

The factor pq^{n-i} is the probability that $h(K_i) = h(K_{n+1})$ and for $i<j\leq n$, $h(K_j) \neq h(K_{n+1})$. Similarly the multiplier of $(n-k)^{\underline{s-k-1}}/(m-k)^{\underline{s-k}}$ on the next line is the probability that $h(K_i) \neq h(K_n)$ for $1\leq i\leq n$ *and* that the first k probes collide. If we now set

$$\begin{aligned}\alpha'_n = & \sum_{1\leq s\leq n} (s-1)\rho_{n,s} \\ = & \; C'_n - 1,\end{aligned}$$

we see that the above recurrence for $\rho_{n,s}$ can be transformed into one for α'_n. In order to perform this manipulation we need the relation

$$\sum_{t<s\leq n} (s-1)(n-t)^{\underline{s-t-1}}/(m-t)^{\underline{s-t}} = m/(m-n)(m-n+1) + (t-1)/(m-n+1),$$

which is an easy consequence of (I1) and (I2). If we interchange summation on s and t we obtain a recurrence for α'_n in terms of α'_i, $i<n$, which can be most easily stated if we use the substitution:

$$\alpha''_n = \alpha'_n - \Big((k-1) - \sum_{1\leq j\leq k} (k-j)\rho_{n,j}\Big) .$$

This aforementioned recurrence is

$$\alpha''_n = (m-n)/(m-n+1) \sum_{1\leq i<n} pq^{n-i}\alpha''_i + (m-k+1)n^{\underline{k}}/(m-n+1)m^{\underline{k}},$$

$$\alpha''_0 = 0.$$

If we now set

$$\beta_n = (m-n+1)\alpha''_n/(m-n)$$

we can subtract two versions of the above recurrence, one for n and one for n-1, to obtain a recurrence in which β_n depends only on β_{n-1}:

$$\beta_n = (1 - 1/(m^{\underline{k}}(m-n+2)))\,\beta_{n-1} + \zeta(m,n) - (1 - 1/(m^{\underline{k}}(m-n+2)))\zeta(m,n-1),$$

$$\beta_0 = 0,$$

where

$$\zeta(m,n) = (m-k+1)n^{\underline{k}}/(m-n)m^{\underline{k}}.$$

This in turn can be simplified a great deal if we use the identity

$$(m-k+1)n^{\underline{k}}/(m-n+1)m^{\underline{k}} + ((k-1) - \sum_{1\le j\le k} (k-j)\rho_{n,j}) = n/(m-n+1),$$

and define

$$\chi_n = \beta_n - \zeta(m,n) = (m-n+1)\,(\alpha'_n - n/(m-n+1))\,/\,(m-n).$$

The final result is contained in the following theorem:

THEOREM 2.4.1. For random k-ary clustering we have

$$C'_n = (m+1)/(m-n+1) + (m-n)\chi_n/(m-n+1), \quad \text{where}$$
$$\chi_n = (1 - 1/(m^{\underline{k}}(m-n+2)))\,\chi_{n-1} + (n-1)^{\underline{k}}/(m^{\underline{k}}m^{\underline{k-1}}(m-n+2)),$$
$$\chi_0 = 0.$$

Asymptotically we have, for k=1 (secondary clustering)

$$C'_\alpha = 1/(1-\alpha) - \alpha - \log(1-\alpha) + O(1/m),$$

while for k-ary clustering with k>1 we have

$$C'_\alpha = \quad 1/(1-\alpha) + O(1/m^{k-1}),$$

as $m \to \infty$, $n = \alpha(m+1)$, fixed $\alpha < 1$.

Thus random tertiary and higher clustering techniques are asymptotically equivalent to uniform probing.

Proof. Note that $(m+1)/(m-n+1)$ is the exact performance of uniform probing as analyzed in Section 1.5. Also note that C'_n equals this value exactly until n exceeds k, for until then no clustering can occur. The argument preceding the Theorem has established the recurrence, so we only have to prove the asymptotic results. For k=1 we have

$$\chi_n = (1 - 1/(m(m-n+2)))\ \chi_{n-1} - 1/m + (m+1)/m(m-n+2)).$$

If we let

$$\psi_n = \psi_{n-1} - 1/m + (m+1)/m(m-n+2))\ (\psi_0 = 0),$$

it is clear that $\psi_n = \chi_n + O(1/m)$ and that

$$\psi_n = -n/m + (m+1)/m\ [H_{m-n+2} - H_{m-2}],$$

from which the assertion of the Theorem for k=1 is clear.

For k>1 the argument that bounds χ_n is even more obvious, showing that this quantity only contributes to the *O* term. ▮

For k = 1 and 2 (and possibly all k) we can obtain a "closed form" expression for C'_n as a finite product. The manipulations are somewhat involved, but the final results are:

for secondary clustering (p = 1/m)

$$C'_n = 1 + m + (m-n)[p/(1-p)] - [(m+1-p)/(1-p)] \prod_{i=1}^{n-1} (1 - p/(m-i+1)),$$

and for tertiary clustering (p = 1/m(m-1))

$$C'_n = 1 + m + (m-n)[p(3-p)/(1-p)(2-p) + pn/(2-p)m]$$

$$- [2(m+1-p)/(1-p)(2-p)] \prod_{i=1}^{n-1} (1 - p/(m-i+1)) .$$

The above two product formulae are due to a suggestion of ([Paterson]).

"breed the best and forget the rest",
cattle ranch advertisement on California route 132

CHAPTER 3:

THE ANALYSIS OF DOUBLE HASHING

This entire chapter is devoted to the analysis of one algorithm, *double hashing*. The technique used is distinctly different from that of the analyses of the introduction and the previous chapter. We can no longer obtain exact recurrence relations for the means. Instead, we prove our asymptotic limit by showing that configurations of occupied positions on the table that would exhibit an average number of probes very different from this limit have negligibly small probability. In spirit the method is mostly akin to the *probabilistic method* of Erdos ([Erd-Spen]). The one fact that makes the argument possible is that the probability of being at least a fixed percent away from the mean in a series of n Bernoulli trials is exponentially small in n. Since arithmetic progressions are the natural probe paths for double hashing, we begin this chapter with a study of the occurrence of arithmetic progressions in random subsets of specified cardinality that are composed of entries in our table.

3.1 The Lattice of Arithmetic Progressions Coming From a Set to a Point

Let Z_m denote the additive group of integers modulo m. We can think of these integers arranged in a circle, with 0 following m-1, as depicted in Figure 3.1.1.

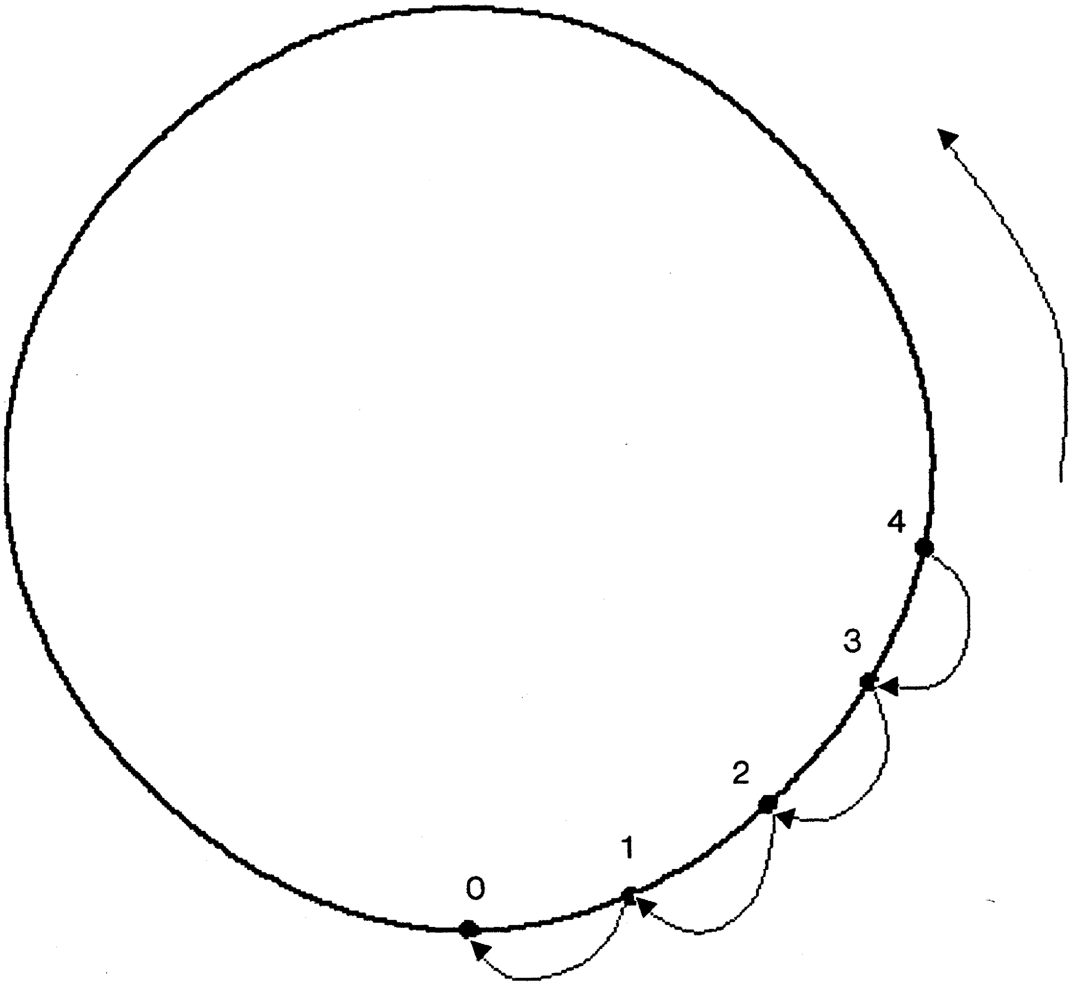
In the entire context of this chapter m is a (sufficiently large) prime number. For any subset $H \subseteq Z_m$ we can count the number of arithmetic progressions that begin at 0 and whose next k elements lie in H. By an arithmetic progression we mean a sequence $x_0, x_1, \ldots, x_k$, such that $x_0 = 0$, $x_1, x_2, \ldots, x_k$ are elements of H, and $x_{i+1} - x_i \pmod{m}$ is the same for all $i = 1,2,\ldots,k-1$. (The point 0 need not itself belong to H.) The primality of m guarantees that all the x_i will be distinct. We will speak of such an arithmetic progression as a progression of length k *coming from* H *to* 0. We can generalize this concept if we allow that for each i, $1 \leq i \leq k$, we specify whether the corresponding element of the progression is to be in H, or in the complement of H. Thus we arrive at the concept of a type of a progression. A *type* τ of length k can be thought of as a boolean vector of k bits. An arithmetic progression of length k coming to 0 is of type τ if the i-th element of the progression is in H or in the complement of H, according to whether the i-th bit of τ is a 1 or a 0. A 1 of the type will also be called a *hit*, whereas a 0 will be called a *miss* (for obvious reasons). We will display a type by writing down the corresponding bit vector, e.g., $\tau = (10110001)$. Any type τ has a *length* that will be usually denoted by k, and a *number of hits*, that will be usually denoted by l, $0 \leq l \leq k$. Thus the above type has $k = 8$, $l = 4$. We will reserve the expression "a progression of length k coming from H to 0" to mean a progression of the type (11...1) with k hits. For any type τ and set H we can consider all the progressions of that type coming to 0. We will speak of these progressions as *belonging* to τ. For a fixed length k, the set of all types of that length forms a boolean lattice (or algebra), in the usual way. Figure 3.1.2. illustrates some of the ordering relationships.

The above lattice structure will not be important immediately, but will play a significant role in the latter half of this chapter.

To fix the ideas let us now confine our attention to arithmetic progressions of length k belonging to the type of all hits. Clearly the number of progressions belonging to this type depends heavily on the set H. We can expect at most to make a probabilistic statement about the distribution of the number of these arithmetic progressions. We will be interested in such an estimation for large m with H of specified cardinality $|H| = \eta m$, where $0 < \eta < 1$. All subsets of this cardinality will be considered equally likely. As we let m get large, we will allow both k and η to vary with m. However, in order to make our argument work, we will see that we have to restrict the growth of k and/or the speed with which η can approach 0

Figure 3.1.1.

THE ADDITIVE GROUP Z_m



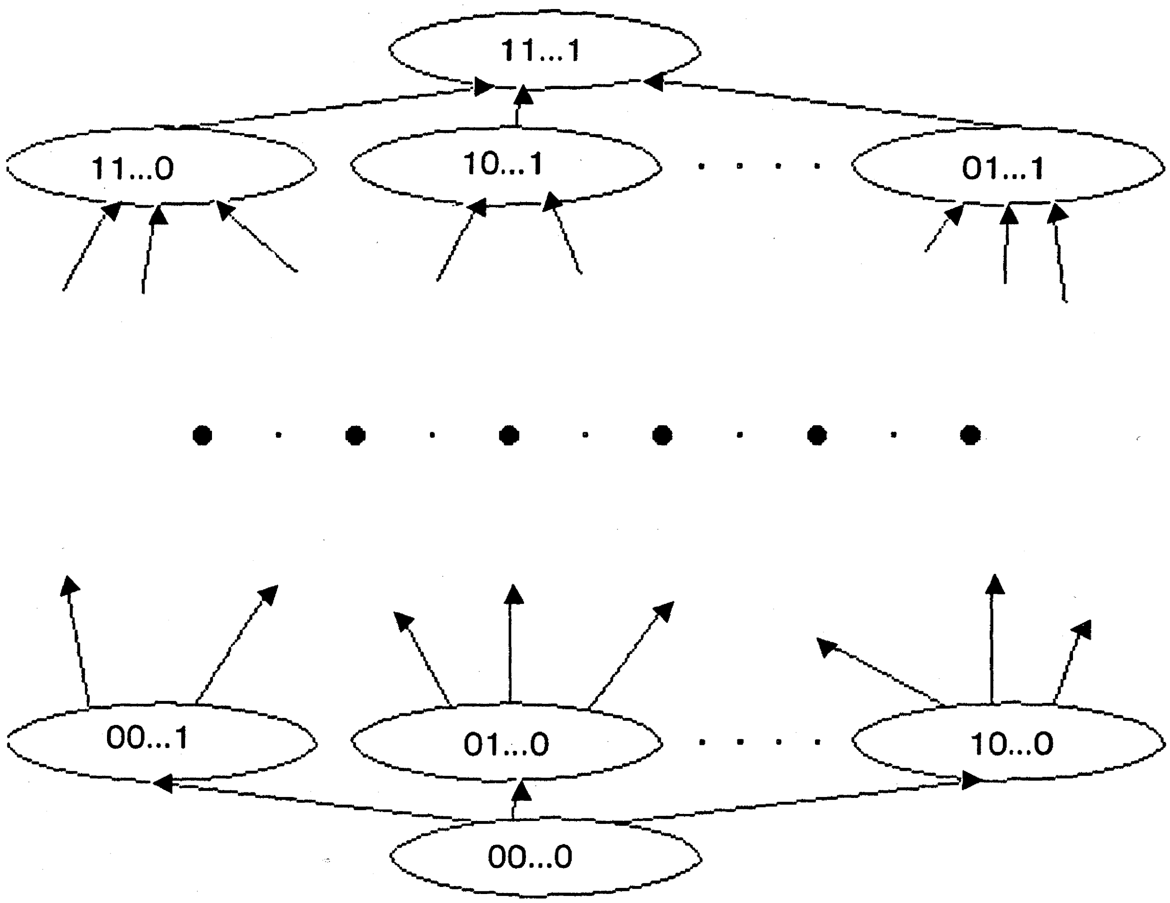

11...1
11...0
10...1
01...1
00...1
01...0
10...0
00...0

Figure 3.1.2. THE LATTICE STRUCTURE OF THE TYPES OF ARITHMETIC PROGRESSIONS

or 1. In this and all subsequent sections, unless otherwise stated, our *O* and *o* notations will always refer to m → ∞. The implied constants, unless otherwise stated, will be absolute.

What is the expected number of arithmetic progressions of length k coming from H to 0? There are m-1 arithmetic progressions in Z_m coming to 0, one for each possible distance 1,2,...,m-1. Each such progression occurs in H with probability

$$\frac{\eta m(\eta m-1) \ldots (\eta m-k+1)}{m(m-1) \ldots (m-k+1)} = (1 + o(1))\ \eta^k,$$

if k is suitably small ($\log k < (\tfrac{1}{2}-\varepsilon) \log m$ will do, though in our applications we will always use a k that is $O(\log m)$) and η is bounded away from 0. Thus the expected number of arithmetic progressions of length k coming from H to 0 is $(1+o(1))\eta^k m$. Let δ denote any small positive constant. In the following three sections we will prove that the fraction of choices of H for which the number of these arithmetic progressions is outside the range $\eta^k m(1\pm\delta)$ is exponentially small in m. By exponentially small we mean that there exist positive constants C, s such that this fraction is

$$\exp[-C\delta^4\eta^k m/(k^s \log^3(\eta^{-k}\delta^{-1}))]$$

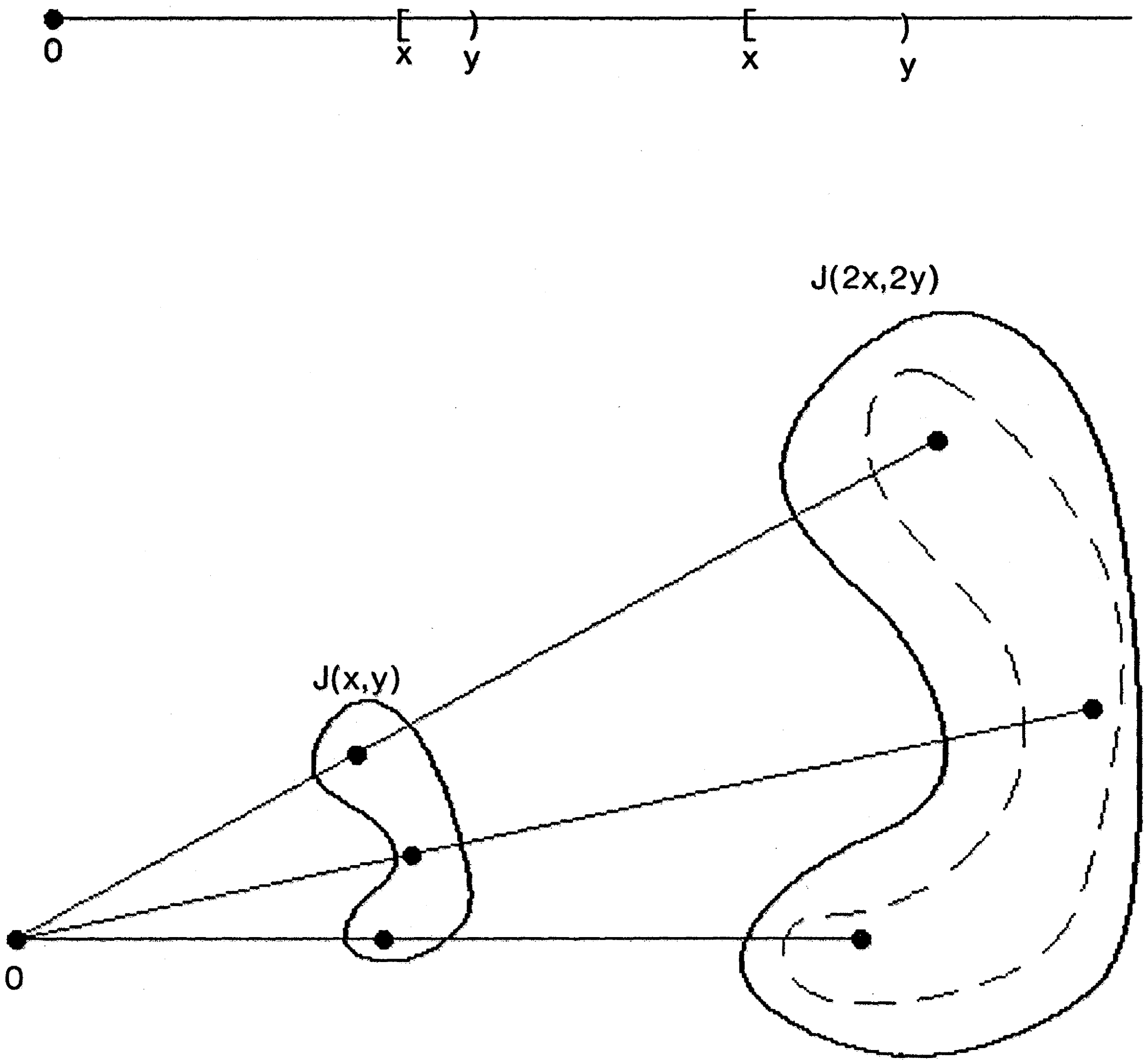
provided $\eta^k m > m^\mu$, where μ denotes any positive constant.

Our method is briefly the following. In Section 3.2 we consider the hypergeometric distribution, which arises when we compute the probability that two subsets of size αm, βm of a set of m elements have an intersection of the expected size $\alpha\beta m$. We show that the probability of the intersection having a size outside the range $\alpha\beta m(1\pm\varepsilon)$ is exponentially small in m. In Section 3.3 we use the Farey series to subdivide the circle formed by the reals (mod 1) into arcs, such that all arcs except those containing certain fixpoints have the property that any two among such an arc's first k multiples are disjoint. By the j-th multiple of an arc (interval) [x,y) we mean the arc [jx,jy) (mod 1). In Section 3.4 we use this subdivision, together with our estimation of the tail of the hypergeometric distribution, to prove the desired result.

The idea of Section 3.4 can be illustrated by an example. Suppose k = 2 and consider an arc [x,y) which is disjoint from the arc [2x,2y), as shown in figure 3.1.3.

Now suppose we pick H, a random set of ηm points of the circle. What is the expected

Figure 3.1.3. AN ILLUSTRATION OF THE "PULL-BACK" ARGUMENT


0
x
y
x
y
J(2x,2y)
J(x,y)
0

number of arithmetic progressions of length 2 coming from H to 0 whose first point lies in the set $J(x,y) = \{z \in Z_m | x \leq (z/m) < y\}$? Instead of repeating the argument given earlier, we can proceed as follows. The interval $J(x,y)$ has size $(y-x)m$. Consider the set $2J(x,y) = \{2z | z \in J(x,y)\} \subseteq J(2x,2y)$. This set also has cardinality $(y-x)m$, and is the locus of the second points of progressions whose first point is in $J(x,y)$. We expect $(y-x)\eta m$ of the points of $2J(x,y)$ to be hit by H. The set of hit points can now be "pulled-back" to $J(x,y)$ to give us the candidate first points of these progressions. Since $J(x,y)$ and $J(2x,2y)$ are disjoint, the expected number of points in this subset of $J(x,y)$ that will be hit is $(y-x)\eta^2 m$. So $(y-x)\eta^2 m$ is the desired average. This argument illustrates how we can translate our knowledge of the probabilities for the size of set intersections to probabilities for the occurrence of arithmetic progressions in H.

All of the above remarks apply verbatim to types other than the type with k hits. If our type has l hits then we only need to replace η^k by $\eta^l(1-\eta)^{k-l}$ everywhere in the above discussion, and restrict η away from both 0 and 1. Circular symmetry implies that our results also hold for any point of Z_m, not just 0. Our method solves the corresponding problem when we do not allow wrap-around or when we specify an upper bound on the number of times we can wrap around. We do not need this result, so we will not dwell on it any longer here. We will, however, need a slight generalization of the case $k = 2$ shown above. Given two points $x,y \in Z_m$, we will say that these points are in the ratio a:b if $xb = ya \pmod m$. Given a fixed ratio a:b, $1 \leq a,b \leq k$, we will want to estimate the number of pairs of points (x,y) of H that are in the ratio a:b.

3.2 The Tail of the Hypergeometric Distribution.

In this section we estimate the tail of the hypergeometric distribution a specified percent away from the mean. Properties of the hypergeometric distribution are discussed, for example, in ([Feller]). Since we are interested in large deviations, the normal approximation will not be useful to us. Instead we will need an approximation more like the one done for the tail of the binomial distribution in ([Renyi]).

Suppose we have a sample space of size n, and we select from this space two subsets, one of size αn, the other of size βn $(0<\alpha,\beta<1)$.

The probability that the cardinality of the intersection of the two subsets is k is

$$a_k = \underbrace{C(\alpha n,k)}_{\text{choose k of } \beta\text{n's from } \alpha\text{n}} \; \underbrace{C((1-\alpha)n,\beta n-k)}_{\text{choose others from rest of n}} \; / \; \underbrace{C(n,\beta n)}_{\text{total number of ways to choose}}.$$

The expected value of k is easily computed to be $\alpha\beta n(1+o(1))$. We will estimate the probability that k lies outside the range $\alpha\beta n\ (1\pm\varepsilon)$. For this section only, our O notation will refer to $n \to \infty$.

THEOREM 3.2.1. Let $Y(n,\alpha,\beta,\varepsilon)$ denote the probability that if we randomly select two sets of sizes αn and βn respectively out of a set of size n, their intersection will have cardinality outside the range $\alpha\beta n(1\pm\varepsilon)$. Then as $n \to \infty$, provided

$$0 < \alpha,\beta \quad \text{and} \quad \alpha(1+\varepsilon),\ \beta(1+\varepsilon) < 1,$$

where α, β, ε can vary with n, we have

$$Y(n,\alpha,\beta,\varepsilon) \quad \leq \quad K(1+1/\varepsilon)e^{-\varphi(\varepsilon)\alpha\beta n},$$

where $\varphi(\varepsilon) \quad \geq \quad (1+\varepsilon)\log(1+\varepsilon)-\varepsilon \ + \ \tfrac{1}{2}\varepsilon^2[\alpha\beta/(1-\alpha)(1-\beta) \ + \ \alpha/(1-\alpha) \ + \ \beta/(1-\beta)]$

and K is an absolute constant.

The same conclusion holds if one of the two sets in question stays fixed.

Proof. We will first estimate the tail of the distribution above the mean. The tail below can be estimated in an essentially identical fashion.

We wish to estimate the sum

$$\sum_{k\geq\alpha\beta n(1+\varepsilon)} a_k \quad = \quad \sum_{k\geq\alpha\beta n(1+\varepsilon)} \quad C(\alpha n,k)\ C((1-\alpha)n,\beta n-k) \ / \ C(n,\beta n).$$

Note that the ratio of two successive terms in this sum is

$$a_{k+1}/a_k \le (\alpha n-k)(\beta n-k)/k((1-\alpha-\beta)n+k),$$

which is a decreasing function of k in the range of interest, i.e., $\alpha\beta n < k < \alpha n, \beta n$.

For $k = \alpha\beta n(1+\varepsilon)$ this ratio is less than

$$\rho = [\ (\alpha-\alpha\beta-\alpha\beta\varepsilon)\ (\beta-\alpha\beta-\alpha\beta\varepsilon)\ ]\ /\ [\ (\alpha\beta+\alpha\beta\varepsilon)\ (1-\alpha-\beta+\alpha\beta+\alpha\beta\varepsilon)\ ].$$

It easily follows that $\rho < 1$. Therefore our sum is majorized by a convergent geometric series of ratio ρ, and we get a bound of

$$[1/(1-\rho)]\ C[\alpha n,\alpha\beta n(1+\varepsilon)]\ C[(1-\alpha)n,(\beta-\alpha\beta(1+\varepsilon))n]\ /\ C(n,\beta n).$$

Since, as we can easily check,

$$1/(1-\rho) = 1 + (1-\alpha-\alpha\varepsilon)(1-\beta-\beta\varepsilon)/\varepsilon \le 1 + 1/\varepsilon,$$

we are only left with estimating the density of the hypergeometric distribution at $k = \alpha\beta n(1+\varepsilon)$, as given above.

We will use Stirling's approximation for the factorial:

$$\log n! = n \log n - n + \tfrac{1}{2}\log n + \tfrac{1}{2} \log 2\pi + O(1/n).$$

From this we can easily derive the following fact:

$$\log C((x+y)n,xn) = (n+\tfrac{1}{2})\ ((x+y)\log(x+y) - x\log x - y\log y) - \tfrac{1}{2}\log n + O(1).$$

We can now apply this fact to the binomial coefficients we have and obtain after simplification

$$\log [\ C(\alpha n,\alpha\beta n(1+\varepsilon))\ C((1-\alpha)n,(\beta-\alpha\beta(1+\varepsilon))n)\ /\ C(n,\beta n)\ ] =$$

$$-[\alpha\beta(1+\varepsilon) \log(1+\varepsilon)$$

$$+ \alpha(1+\beta)\{1-[(\beta\varepsilon)/(1-\beta)]\} \log\{1 - [(\beta\varepsilon)/(1-\beta)]\}$$

$$+ \beta(1-\alpha)\{1-[(\alpha\varepsilon)/(1-\alpha)]\} \log\{1 - [(\alpha\varepsilon)/(1-\alpha)]\}$$

$$+(1-\alpha)(1-\beta)\{1 + [(\alpha\beta\varepsilon)/(1-\alpha)(1-\beta)]\} \log\{1 + [(\alpha\beta\varepsilon)/(1-\alpha)(1-\beta)]\}]\ n$$

$$+\ O(1).$$

The following two inequalities are elementary:

$$(1+x)\log(1+x) \geq x \text{ for } x \geq 0,$$

$$(1-x)\log(1-x) \geq -x + (x^2/2) \text{ for } 0 \leq x \leq 1.$$

Since $\alpha(1+\varepsilon)<1$ is equivalent to $\alpha\varepsilon/(1-\alpha)<1$, and similarly for β, we can apply these inequalities to the above expression to obtain the upper bound

$$-[\alpha\beta\varepsilon + \alpha\beta[(1+\varepsilon)\log(1+\varepsilon) - \varepsilon]$$

$$-\alpha\beta\varepsilon + \alpha\beta^2\varepsilon^2/2(1-\beta)$$

$$-\alpha\beta\varepsilon + \alpha^2\beta\varepsilon^2/2(1-\alpha)$$

$$+\alpha\beta\varepsilon]n,$$

from which the conclusion of the theorem is immediate.

For the lower tail a similar argument gives an upper bound of

$$-[\alpha\beta[(1-\varepsilon)\log(1-\varepsilon)+\varepsilon] + \tfrac{1}{2}\alpha^2\beta^2\varepsilon^2/(1-\alpha)(1-\beta)]\ n + O(1).$$

Now $(1-\varepsilon)\log(1-\varepsilon)+\varepsilon \geq (1+\varepsilon)\log(1+\varepsilon)-\varepsilon$ and the theorem follows.▮

Remark 1. Notice that the above argument does not require ε to be small.

Remark 2. If, however, ε is small, say $\varepsilon \leq \varepsilon_0$, then $\varphi(\varepsilon) \geq (1+\varepsilon)\log(1+\varepsilon)-\varepsilon \geq C\varepsilon^2$ where C depends on ε_0. If $\alpha\beta n\varepsilon^2/\log(1/\varepsilon) > N$, where N is a sufficiently large constant, then we can take C so small that the factor $K(1+1/\varepsilon)$ is absorbed in the reduced exponent. Then we can state our conclusion as

COROLLARY 3.2.1

$Y(n,\alpha,\beta,\varepsilon) \leq \exp(-C\varepsilon^2\alpha\beta n)$ for $\varepsilon\leq\varepsilon_0$, $\alpha\beta n\varepsilon^2/\log(1/\varepsilon) > N$, N and C positive constants depending on ε_0.

This is the form in which theorem 3.2.1 will be used most often. In our applications in fact, $\alpha\beta n\varepsilon^2/\log(1/\varepsilon)$ will tend to ∞ with n.

The key property of our estimate is that it is exponentially small in n. An estimate obtained by using the variance and Chebycheff's inequality can only give us a bound for this tail that vanishes no faster than an inverse power of n.

3.3. The Farey Subdivision of the Circle.

The Farey series F_n of order n is the ascending series of irreducible fractions between 0 and 1 whose denominators do not exceed n. For example, F_5 is

0/1, 1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 1/1.

The Farey series possesses many fascinating properties ([Hardy]).

Property 1. If h/k, h'/k' are two successive terms of F_n, then kh'-hk' = 1.

Property 2. If h/k, h"/k", and h'/k' are three successive terms of F_n, then

h"/k" = (h+h')/(k+k').

Property 3. If h/k, h'/k' are two successive terms of F_n, then k+k'>n.

Property 4. If n>1, then no two successive terms of F_n have the same denominator.

Property 5. The number of terms in the Farey series of order n is asymptotically $3n^2/\pi^2 + O(n\log n)$.

We will be interested in the circle of the reals (mod 1), denoted by U. The set U forms a group under addition. Consider the mappings

$$T_i: x \rightarrow ix \pmod 1,\ x \in U$$

for each i = 2,3,...,k. (T_1 is the identity.) It should be clear that the *fixpoints* (i.e., points x for which $T_j x = x$ for some j) of these mappings are the fractions a/b with $0 \leq a < b < k$. These are exactly the elements of $F_{k-1} \subset U$.

We wish now to partition U into a collection of disjoint intervals (taken to be left closed, right open) J with the property that (1) if $V \in J$, then $T_1V, T_2V, T_3V, \ldots, T_kV$ are all disjoint if V does not contain one of the above fixpoints, and (2) the $V \in J$ that contain a fixpoint can be made arbitrarily small in length.

Let φ_n denote (the cardinality of F_n)-1. We now consider the subdivision of the circle defined by the Farey series F_{2k-2}. Clearly this contains the fixpoints discussed above ($F_{k-1} \subset F_{2k-2}$) and subdivides the circle into φ_{2k-2} intervals. We have

LEMMA 1. No two fixpoints (i.e., elements of F_{k-1}) are adjacent in F_{2k-2}.

Proof. If fixpoints h_1/k_1, h_2/k_2 are adjacent in F_{k-1}, then $k_1+k_2 \leq 2k-2$. Hence by Property 3 they cannot be adjacent in F_{2k-2}. ∎

For i = 1,2,...,φ_{k-1}, let us name L_i and R_i respectively the intervals of the above subdivision that lie to the "left" and to the "right" of the i-th fixpoint (in the standard order).

From Lemma 1 it follows that the other endpoint of each L_i and R_i is not a fixpoint. We name the remaining intervals N_i, i = 1,2,...,($\varphi_{2k-2}-2\varphi_{k-1}$).

LEMMA 2. Let X stand for one of the L_i or R_i. Then any two of

$$T_1X,\ T_2X,\ \ldots,\ T_kX$$

will overlap only if they have an endpoint in common.

Proof. Let the i-th fixpoint be h_1/k_1 and to make things concrete suppose we are dealing with R_i (the case of L_i is entirely analogous). Let h_2/k_2 be the other endpoint of R_i. Then by Lemma 1, $h_2/k_2 \notin F_{k-1}$, so $k_2 \geq k$. Then the length of R_i is

$$(h_2/k_2) - (h_1/k_1) = (1/k_1k_2) \leq (1/kk_1).$$

Thus the length of any of the T_jR_i is $\leq(1/k_1)$. But all multiples of $(h_1/k_1, h_2/k_2)$ start at a multiple of h_1/k_1. These multiples are spaced at least $1/k_1$ apart, so no two of the multiples of R_i will overlap unless they share a common left endpoint. ▮

LEMMA 3. Let Y denote one of the N_i. Then the intervals

$$T_1Y,\ T_2Y,\ \ldots,\ T_kY$$

are all disjoint.

Proof. Suppose intervals A and B in the above sequence intersect. By construction A and B do not share any endpoints. Thus if they intersect, we can assume that the left endpoint of B lies within A. Let Y be $(h_1/k_1, h_2/k_2)$. The distance between the left endpoints of A and B cannot be less than $1/k_1$, since the left endpoints are multiples of $1/k_1$. However even the longest multiple has only length $k(h_2/k_2 - h_1/k_1) = k/k_1k_2 \leq 1/k_1$, since $k_2 \geq k$ as no endpoint is a fixpoint. But this contradicts our assumption that the left endpoint of B lies within A. ▮

We now describe how to subdivide the L_i and R_i further, so as to make the intervals with an endpoint at a fixpoint as small as we please, while still maintaining the property that all other intervals have disjoint multiples. We describe the construction for R_i, that for L_i being analogous. Let us define for each i a subdivision into intervals RS_{ij}, $j = 1,2,\ldots,l$, and RM_i. If $R_i = [x,y)$, these subintervals are defined as follows:

$$RS_{ij} = [x+((k-1)/k)^j(y-x),\ x+((k-1)/k)^{j-1}(y-x)),\ j = 1,2,\ldots,l$$

$$RM_i = [x,\ x+((k-1)/k)^l(y-x)).$$

The following facts are then obvious:

(1) $RM_i \cup \bigcup_{j=1}^{l} RS_{ij} = R_i$,

(2) any two of RS_{i1}, RS_{i2}, ..., RS_{il}, RM_i are disjoint,

(3) RS_{ij} has length $((k-1)/k)^{j-1}(y-x)/k$, and RM_i has length $((k-1)/k)^l(y-x)$ ($y-x$ = length of R_i).

LEMMA 4. Let Y denote any of the RS_{ij}. Then the intervals

$$T_1Y,\ T_2Y,\ \ldots,\ T_kY$$

are disjoint.

Proof. We first prove the lemma for $i = 1$, i.e., for a subdivision of the interval R_1 whose left endpoint is 0 (= 1). As $1/k \in F_{2k-2}$, $R_1 \subset [0,1/k)$ so we don't have to worry about wrap-around problems for any of the RS_{1j}. To complete our argument we only need show that the right endpoint of the (t-1)-st multiple does not exceed the left endpoint of t-th multiple. This is tantamount to

$$(t-1)((k-1)/k)^{j-1} \leq t((k-1)/k)^j$$

or

$$(t-1)/t \leq (k-1)/k,$$

which is certainly true, as t only takes the values 1,2,...,k. This subdivision is nicely illustrated by Figure 3.3.1.

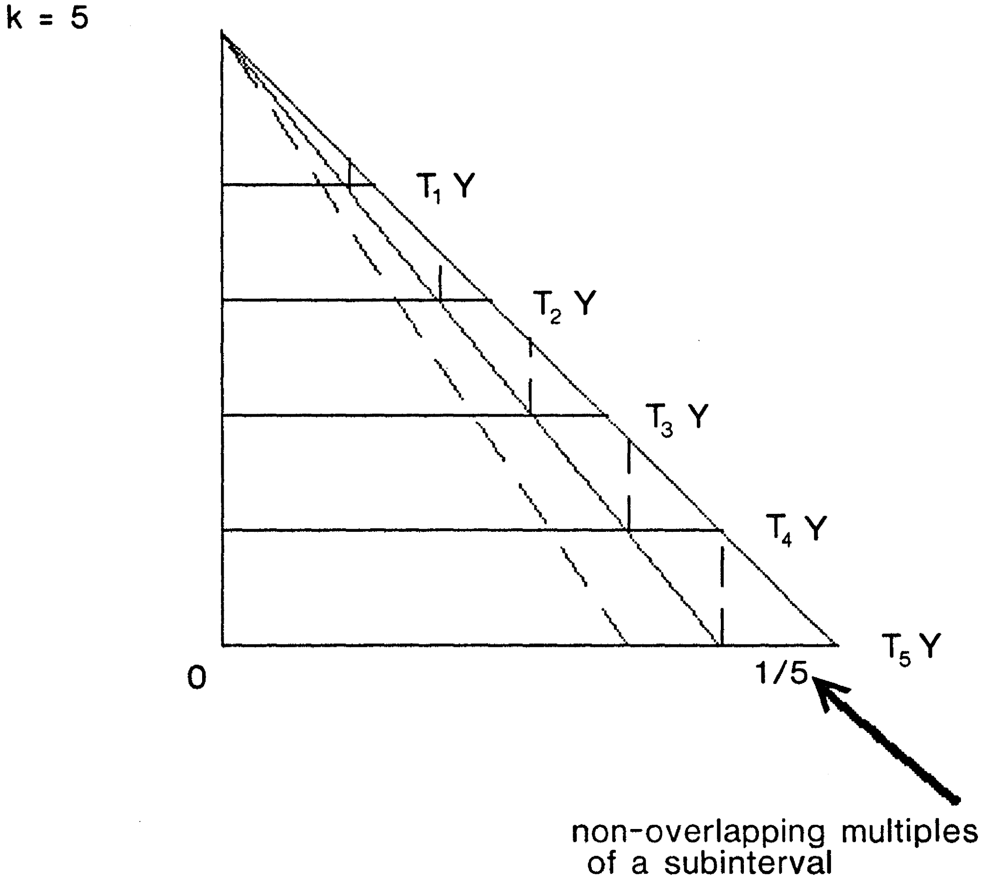
Now to handle the case of $i>1$ we need only recall from Lemma 2 that two multiples of R_i overlap only if they have a common left endpoint. But then the situation at each such endpoint is a subcase of the situation described above for $i=1$ around 0. So by the same argument the multiples of RS_{ij} are disjoint. ∎

We can of course repeat the whole construction and the proof of Lemma 4 for the L_i. Thus we obtain intervals LS_{ij}, $j = 1,2,\ldots,l$, and LM_i that also satisfy (1), (2), and (3) above.

Before we recapitulate what we have derived in this section we need to make a comment about the lengths of the intervals. Each R_i or L_i, being an interval $[h_1/k_1, h_2/k_2)$ between

Figure 3.3.1.

THE SUBDIVISION INTO INTERVALS OF NON-OVERLAPPING MULTIPLES NEAR A FIXPOINT


k = 5
T_1 Y
T_2 Y
T_3 Y
T_4 Y
T_5 Y
0
1/5
non-overlapping multiples of a subinterval

two elements of F_{2k-2}, has length $1/k_1k_2 \geq 1/4k^2$. On the other hand either k_1 or k_2 is $\geq k$, so the length is at most $1/k$.

Combining all our constructions we have the following theorem.

THEOREM 3.3.1. We can construct a partition of the reals mod 1 into disjoint subintervals

N_i, $i = 1,2,\ldots,\varphi_{2k-2}-2\varphi_{k-1}$,

LS_{ij}, LM_i, RS_{ij}, RM_i, $i = 1,2,\ldots,\varphi_{k-1}$ $j = 1,2,\ldots,l$

so that

1. each of the N_i, LS_{ij}, RS_{ij} has (a) disjoint first k multiples and (b) length at least $((k-1)/k)^{l-1}/4k^2$, and

2. each of the LM_i, RM_i has (a) an endpoint (the right or left one, respectively) which is a fixpoint and (b) length at most $((k-1)/k)^l/k$.

3.4. The Estimation of the Arithmetic Progressions, and the Prevalence of Randomness.

We first map intervals on U to intervals on Z_m. Corresponding to an interval $[x,y) \subset U$ we have the set of all $i \in Z_m$ with the property that $x \leq (i/m) < y$. (This should be interpreted cyclically; that is, if $x > y$, then we mean $x \leq i/m$ *or* $i/m < y$). We will now use the names of the intervals introduced in Section 3.3 to denote also the corresponding intervals in Z_m. For an interval $T = [x,y)$ we will denote by t its length in U ($t = (y-x)$) and by $|T|$ the number of integers it contains in Z_m. Clearly we have $|T|$ = (number of i such that $xm \leq i < ym$) $= \lceil ym \rceil - \lceil xm \rceil$. Thus

$$|T| = tm \pm 2.$$

We will write this as

$$\text{(i)} \quad |T| = tm(1 \pm \varepsilon),$$

where ε will be a quantity used below. This is justified as long as ε is not too small. As we will see below, the smallest ε we will use will be $\Omega(1/(\log m)^r)$ for some positive r, while the smallest t will be such that $tm \geq m^\lambda$ for some positive λ. So as $m \rightarrow \infty$, equation (i) is justified; its form will simplify some of the computation below.

Recall that we are selecting a random subset H of Z_m of cardinality ηm. For any interval (in fact any subset) T of Z_m, we can ask for the number of elements of H that will fall in T. From Theorem 3.2.1 we know that the number of these elements will be $|T|\eta(1\pm\varepsilon)$, except with probability $Y(m,|T|/m,\eta,\varepsilon)$, i.e., it will be $\eta tm(1\pm\varepsilon)^2$ except with probability $Y(m,t(1\pm\varepsilon),\eta,\varepsilon)$.

Consider now the collection *D* of intervals composed of all the N_i, LS_{ij}, RS_{ij}, and the collection *C* composed of these same intervals *and* their first k multiples. In this latter case we are dealing with a total of $k(2\varphi_{k-1}l+\varphi_{2k-2}-2\varphi_{k-1})$ intervals. For each interval T in the collection *C* we assume that H will intersect it in $\eta tm(1\pm\varepsilon)^2$ elements, as in the discussion above. This will always be the case except for a fraction of choices of H that is bounded by

$$Q = \sum_{T \varepsilon C} Y(m,t(1\pm\varepsilon),\eta,\varepsilon).$$

(This argument does not need any independence assumptions concerning the various choices of T.) Thus with probability 1-Q, our choice of H will intersect each interval in the collection about as often as we expect.

We now restrict T to be one of the elements of *D*. In the sequel ε will denote a small quantity that will define all our relative errors. We allow $\varepsilon \rightarrow 0$ as $m \rightarrow \infty$, and we also allow ε to depend on our choice for T. (We write ε_T when we need to make this dependency explicit.) Let us consider the first k multiples of T, and focus our attention on the last one T_kT. This interval has $tkm(1\pm\varepsilon)$ elements, and will almost certainly receive $tk\eta m(1\pm\varepsilon)^2$ elements of H. Within T_kT we have a subset S_k of cardinality $tm(1\pm\varepsilon)$, consisting of those elements that are k-fold multiples of elements of T. How many elements of this subset will H hit? (Note that these points are the endpoints of arithmetic progressions starting at 0, having their first element in T,..., and their k-th in T_kT.) Now within T_kT itself we can invoke Theorem 3.2.1 to show that, except for a fraction of possibilities that does not exceed $Y(tkm(1\pm\varepsilon),(1/k)(1\pm\varepsilon)^2,\eta(1\pm\varepsilon)^3,\varepsilon)$ the number of elements of S_k that H will hit will be $\eta tm(1\pm\varepsilon)^7$.

Consider now the $\eta tm(1\pm\varepsilon)^7$ progressions thus specified. We apply the "pull-back" process illustrated in Section 3.1. and in the following Figure 3.4.1. What about the (k-1)-st points of these progressions -- how many of these points will be hit by H? By construction, all these points from a subset S_{k-1} of $T_{k-1}T$, an interval *disjoint* from T_kT. By Theorem 3.2.1 confined now to the interval $T_{k-1}T$ we see that the intersection of S_{k-1} and H will be $\eta^2 t(1\pm\varepsilon)^{13}m$ points, except with probability $Y(t(k-1)m(1\pm\varepsilon),(\eta/(k-1))(1\pm\varepsilon)^8,\eta(1\pm\varepsilon)^3,\varepsilon)$. (To amplify, we have here an interval of $t(k-1)m(1\pm\varepsilon)$ points; the set S_{k-1} is of size $(\eta/(k-1))(1\pm\varepsilon)^8$ times the size of $T_{k-1}T$; and $\eta(1\pm\varepsilon)^3$ is the probability that a point in $T_{k-1}T$ will be hit by H. The basic rule we are using throughout is that if $x = X(1\pm\varepsilon)^i$, $y = Y(1\pm\varepsilon)^j$, then $xy = XY(1\pm\varepsilon)^{i+j}$, $x/y = (X/Y)(1\pm\varepsilon)^{i+j}$. To make these rules precise it is best to define $x = X(1\pm\varepsilon)$ to mean $x \in (X(1-\varepsilon), X/(1-\varepsilon))$. Note that this redefinition of $1\pm\varepsilon$ leaves Theorem 3.2.1.valid.)

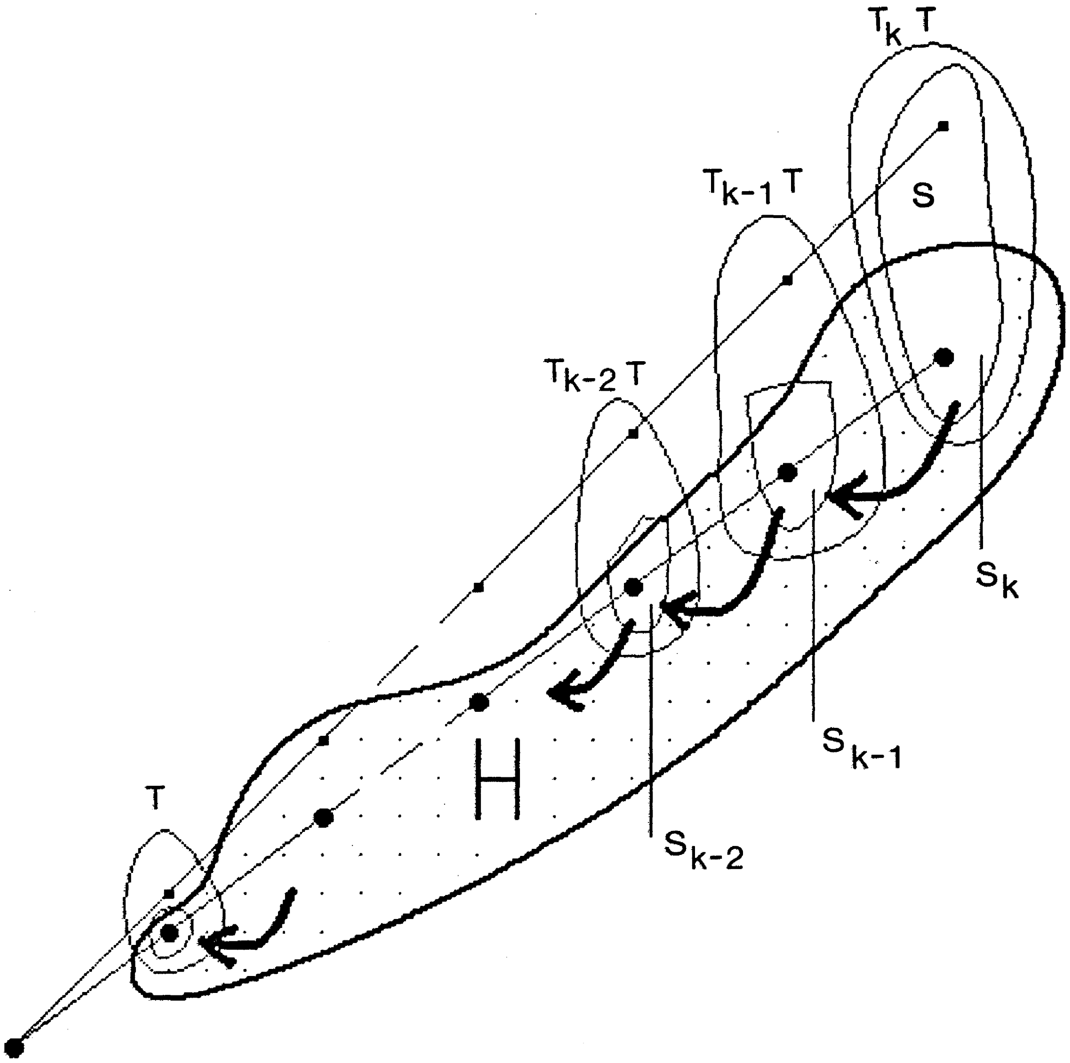
We now have a set of progressions whose last two elements are guaranteed to be hit by H. At the next step we consider the (k-2)-nd points of these $\eta^2 tm(1\pm\varepsilon)^{13}$ progressions, which define a subset S_{k-2} of $T_{k-2}T$. By analogous computation we obtain that $\eta^3 tm(1\pm\varepsilon)^{19}$ of these points will belong to our random set H, except with probability $Y((k-2)tm(1\pm\varepsilon),(\eta^2/(k-2))(1\pm\varepsilon)^{14},\eta(1\pm\varepsilon)^3,\varepsilon)$. We now continue in this fashion with the (k-3)-rd,...,1-st points of the arithmetic progressions. The illustration 3.4.1. depicts this *pull-back* process in which we successively commit H in the intervals T_iT, i = k,k-1,...,1. At the last step of this process we are considering T itself. After that step the number of candidate arithmetic progressions left will be $\eta^k tm(1\pm\varepsilon)^{6k+1}$. These are now confirmed to be entirely in H. The fraction of choices for H that we have eliminated in this process is bounded by the sum

$$\sum_{i=0}^{k-1} Y((k-i)tm(1\pm\varepsilon),\ (\eta^i/(k-1))(1\pm\varepsilon)^{6i+2},\ \eta(1\pm\varepsilon)^3,\ \varepsilon)$$

of all the excluding probabilities.

We now conceive of this process of selection of candidate arithmetic progressions as being carried out for T referring in turn to each of the intervals in *D*. The total fraction of choices for H thus excluded is bounded by

Figure 3.4.1. THE PULL-BACK PROCESS


$T_k T$
S
$T_{k-1} T$
$T_{k-2} T$
S_k
S_{k-1}
H
T
S_{k-2}

$$W = \sum_{T \in D} \sum_{i=0}^{k-1} Y((k-i)tm(1 \pm \varepsilon_T),\ (\eta^i/(k-1))(1 \pm \varepsilon_T)^{6i+2},\ \eta(1 \pm \varepsilon_T)^3,\ \varepsilon_T).$$

What has all this accomplished? After excluding the choices for H accounted for in Q and W, we can be sure that the number of arithmetic progressions of length k coming from H to 0, and whose first point is T, is $\eta^k tm(1 \pm \varepsilon_T)^{6k+1}$, where T is any of the above intervals. Thus the total number of arithmetic progressions coming to 0 from H of length k is

$$\sum_{T \in D} \eta^k tm(1 \pm \varepsilon_T)^{6k+1} + E$$

where E is a correction coming from the fact that we cannot apply our argument to the fixpoint intervals LM_i and RM_i. But each of these special intervals cannot contribute more arithmetic progressions than its length. Thus the error committed is bounded by

$$0 \leq |E| \leq 2\varphi_{k-1}[(1/k)((k-1)/k)^l m + 2].$$

(See Theorem 3.3.1 and recall that there are $2\varphi_{k-1}$ such intervals).

Now let δ be a given small positive constant. We choose l to be the smallest positive integer such that

(ii) $$2\varphi_{k-1}(1/k)((k-1)/k)^l m \leq \eta^k m\delta/4.$$

Thus $l = \lceil(k\log(1/\eta)+\log\varphi_{k-1}-3\log 2+\log(1/\delta))/(\log((k-1)/k))\rceil$. By choosing δ small enough, and using the fact that $|\log(1-1/k)| \geq C/k$ for $k>1$ and some positive constant C, we see that l will always satisfy

(iii) $$l \geq 2k.$$

Since $\eta^k m > m^\mu$ it is also clear that for m sufficiently large

$$(1/k)((k-1)/k)^l m \geq 2,$$

and so

(iv) $|E| \leq 2\varphi_{k-1}[(1/k)((k-1)/k)^l m+2] \leq \eta^k m\delta/2.$

The total number of intervals in our partition is then

$$F = 2\varphi_{k-1}(l+1)+\varphi_{2k-2}-2\varphi_{k-1} = \varphi_{2k-2}+2l\varphi_{k-1}.$$

If t denotes the length of an interval in our collection *D*, then from Theorem 3.3.1. we have

$$t \geq ((k-1)/k)^{l-1}/4k^2 \geq \eta^k\delta/(32\varphi_{k-1}k(k-1))$$

$$\geq S\eta^k\delta/k^4 \text{ for some positive constant S.}$$

We can therefore write

(v) $S\eta^k\delta/k^4 \leq t \leq 1/k.$

(See Properties 3 and 5 of Section 3.3).

For an interval $T \in D$ of length t, let ε_T be defined by

$$1/(1-\varepsilon_T) = (1 + \delta/(2tF))^{1/(6k+1)},$$

so we assign larger relative errors to smaller intervals.

Now we are ready to total the number of arithmetic progressions we have of length k coming from H to 0. This number cannot exceed

$$\eta^k m(\delta/2) + \sum_{T \varepsilon D} \eta^k mt(1+\delta/(2Ft)) \leq \eta^k m(\delta/2) + \eta^k m + \eta^k m(\delta/2)$$

$$\leq \eta^k m(1+\delta).$$

In order to obtain a lower bound we use the elementary inequality

$$(1+x)^{-y} \geq (1-x)^y \text{ for } 1 \geq x,y \geq 0.$$

Then we have

$$1-\varepsilon_T = (1 + \delta/2tF)^{-1/(6k+1)} \geq (1 - \delta/2tF)^{1/(6k+1)}.$$

We must avoid, however, situations where $\delta/2tF$ comes too close to 1. We stipulate therefore that we will not attempt to maintain a lower bound during the pull-back process for an interval T, unless $tF \geq \delta$. Thus we will ignore lower bounds for intervals T much smaller than the average (the average length of an interval is $1/F$). As our intervals form sequences with lengths in geometric progressions, we expect that the total length of the uncontrolled intervals (we include in this the LM_i and RM_i) will be small. First of all it is easy to see that if δ is small enough then none of the intervals N_i will violate the condition $tF \geq \delta$. In each sequence the total length of the intervals violating our condition is certainly less than

$$(\delta/F) \sum_{i=0}^{\infty} ((k-1)/k)^i = \delta k/F$$

for a total contribution not exceeding

$$\delta \; 2\varphi_{k-1}k/F.$$

But we have $F \geq 2l\varphi_{k-1} \geq 4k\varphi_{k-1}$ by (iii), and so the total length of the uncontrolled intervals does not exceed $\delta/2$. Therefore the total of progressions we are counting is at least

$$\sum_{\substack{T \in \mathcal{D} \\ tF \geq \delta}} \eta^k mt(1-(\delta/2tF)) \geq (1-(\delta/2))\eta^k mt - \eta^k m(\delta/2) = \eta^k m(1-\delta).$$

In summary, the total of our progressions is

$$\eta^k m(1 \pm \delta),$$

as we had hoped to prove. (Note again $1+\delta < 1/(1-\delta)$).

This, of course, is a useful result only if we can show that the sum $Q+W$ of the excluding probabilities is small. To prove this we will need to restrict our η and k. We will assume that

$$\eta^k m \geq m^{\mu}, \; k = O(\log m)$$

where μ is any small positive constant. We now show that each term in Q or W is exponentially small in m. We will determine an upper bound for the largest term, which certainly occurs in W. A candidate term has the form

$$\exp(-\varphi(\varepsilon_T)(1\pm\varepsilon_T)^{(6i+6)/(6k+1)}\eta^{i+1}tm).$$

We first treat the 1 - ε_T case. For any interval T we are attempting to control we have

$$1-\varepsilon_T \geq (1-(\delta/2tF))^{1/(6k+1)} \geq (1/2)^{1/(6k+1)}.$$

Thus the absolute value of the above exponent is at least

$$\varphi(\varepsilon_T)(1/2)(1/2)^{5/(6k+1)}\eta^{k+1}(\delta/F)m.$$

For the case $1+\varepsilon_T$ we have

$$(1+(\delta/2tF))^{(6i+6)/(6k+1)}\eta^{i+1}tm \geq (1+(\delta/2tF))^{(6i+6)/(6k+6)}\ \eta^{i+1}\ t\ m$$

$$\geq (\delta/2F)^{(6i+6)/(6k+6)}\ t^{6(k-i)/(6k+6)}\ \eta^{i+1}\ m$$

$$\geq (\delta/2F)\ t^{6(k-i)/(6k+6)}\ \eta^{i+1}\ m,$$

as certainly we can take $\delta<1$. Let now t have its smallest value $S\eta^k\delta/k^4$ (from (v)) and obtain a lower bound of

$$S\ (\delta^2/(2Fk^4))\eta^{(6k(k-i)/(6k+6))+i+1}m \geq (S\delta^2/(2Fk^4))\eta^{k+1}m.$$

Thus the largest term does not exceed

$$\exp(-\varphi(\varepsilon_T)(S\delta^2/2Fk^4)\eta^{k+1}m).$$

Finally for $\varphi(\varepsilon_T)$ we have

$$\varphi(\varepsilon_T)\geq(1+\varepsilon_T)\log(1+\varepsilon_T)-\varepsilon_T$$

by Theorem 3.2.1; note also that (1+x)log(1+x)-x is an increasing function of x. Now

$$\varepsilon_T = 1 - (1 + (\delta/2tF))^{-1/(6k+1)} \geq$$

$$1 - (1 + (\delta k/2F))^{-1/(6k+1)}.$$

Since $1-(1+x)^{-1/p} \geq c(x/p)$ if $0<x<\rho<1$, where p is an integer and c a constant depending on ρ, we can conclude

$$\varepsilon_T \geq (c_1\delta/F),$$

for some positive constant c_1, if δ is small -- say $\delta<1/2$. Therefore

$$\varphi(\varepsilon_T) \geq \varphi(c_1\delta/F) \geq (c_2\delta^2/F^2)$$

for some other positive constant c_2. Combining all of the above we see that there exists a positive constant c_3 such that no term of Q or W exceeds

$$\exp(-c_3\delta^4\eta^{k+1}m/k^4F^3).$$

From the definition of l we see that

$$l = O(k^2\log(1/\eta)+k\log k+k\log(1/\delta)), \quad F = O(k^2l)$$

where the implied constants are absolute. We can finally find an absolute positive constant C that incorporates also the effect of these *O*-constants and that of adding all the terms in Q and W. For that constant C we can then conclude that

$$Q+W \leq \exp(-C\delta^4\eta^{k+1}m/[k^{16}(k\log 1/\eta+\log k+\log 1/\delta)^3]).$$

We are now basically done. It only remains to check that the assumptions we used were justified. It is very easy to check that assumption (i) is satisfied for the values of ε_T we have chosen. For our repeated applications of the set intersection theorem we need to know that

$$t/(1-\varepsilon_T) < 1, \quad \eta/(1-\varepsilon_T)^4 < 1.$$

Now

$$t/(1-\varepsilon_T) = t(1+(\delta/2tF))^{1/(6k+1)} \leq t\,(1 + \delta/(2(6k+1)Ft))$$

$$\leq t + \delta/(2(6k+1)F) < 1 \text{ since } t<1/2,\ \delta<1.$$

Now note that when we choose ε_T we can assume that $\eta^k/(1-\varepsilon_T)^{6k+1} \leq t$, for certainly we cannot have more progressions than the length of T. Thus to check $\eta/(1-\varepsilon_T)^4 < 1$ we look at $\eta^{6k+1}/(1-\varepsilon_T)^{4(6k+1)}$. Now

$$\eta^{6k+1}/(1-\varepsilon_T)^{4(6k+1)} = (\eta^k/(1-\varepsilon_T)^{6k+1})^4 \eta^{2k+1} < \eta^{2k+1} < 1.$$

Thus $\eta/(1-\varepsilon_T)^4 < 1$ is also proved. This completes the argument for the following result:

THEOREM 3.4.1. If μ, δ_0 are positive constants while η, δ, and k can vary with m so that

$$0 < \eta < 1,$$
$$1 \leq k = O(\log m),$$
$$\eta^k m \geq m^\mu,$$
$$0 < \delta \leq \delta_0,$$

then there exists a constant C depending at most on δ_0 such that as $m \to \infty$, except with probability not exceeding

$$\exp(-C\delta^4\eta^{k+1}m/[k^{16}(k\log 1/\eta+\log k+\log 1/\delta)^3]),$$

a selection H of ηm points in Z_m will have

$$\eta^k m(1\pm\delta)$$

arithmetic progressions of length k coming from H to 0.

THEOREM 3.4.2. If τ is a type of length k and l hits, then Theorem 3.4.1 applies to the enumeration of progressions of type τ coming to 0, if throughout we replace η^k by $\eta^l(1-\eta)^{k-l}$. That is under the assumption that

$$\eta^l(1-\eta)^{k-l}m \geq m^\mu,$$

we can conclude that the number of arithmetic progressions of type τ coming from H to 0 will be

$$\eta^l(1-\eta)^{k-l}m(1\pm\delta),$$

except with probability not exceeding

$$\exp(-C\delta^4\eta^{k+1}(1-\eta)^{k-l+1}m/[k^{16}(k\log(\eta^{-1}(1-\eta)^{-1})+\log k+\log 1/\delta)^3]).$$

Proof. In the argument above intersect with H or the complement of it according to whether the type specifies a hit or a miss. ▮

CORROLARY 3.4.1. The conclusion of Theorem 3.4.1 or Theorem 3.4.2 can be made to apply *simultaneously* for all progressions coming to all points x in Z_m and all types not exceeding a certain length $k_0 = O(\log m)$.

Proof. Simply look at the sum of all the excluding probabilities. The total number of conditions we are imposing is a polynomial in m (e.g., <number of types> x <number of points>). Now use the fact that $P(m)\exp(-C_1 m^{\mu}) < \exp(-C_2 m^{\mu})$ as $m \to \infty$ if P denotes a polynomial and $C_2 < C_1$. ∎

This last corollary illustrates the power of the exponentially small bounds.

COROLLARY 3.4.2. Under the conditions of Theorem 3.4.1, a selection H of ηm points in Z_m will have no more than

$$2\eta^2 m$$

pairs (x,y), $x,y \in H$, of points in the specified ratio a:b, $1 \leq a,b \leq k$, except with probability not exceeding

$$\exp(-C\eta^{k+1} m/[k^{16}(k\log 1/\eta + \log k)^3]).$$

Proof. Assume the ratio a:b is in lowest terms. Then apply the argument of this section while only considering T_aT and T_bT in the pull-back process for each interval T. ∎

The following generalization of Corollary 3.4.2 is needed in Section 3.8. The arbitrary notation used below is chosen to correspond to the context of that section.

THEOREM 3.4.4. Let A denote a fixed subset of Z_m of cardinality at least $m^{1/4+\delta_2}$. Let $\eta = m^{-1/4-\delta_1}$, where δ_1, δ_2 are small positive constants satisfying $\delta_2 > \delta_1$. Then there exists a small positive constant δ such that: if a subset H of ηm elements is randomly chosen in Z_m, then the number of pairs of points (u,v) in H with $v \in A$, and u, v in the prespecified ratio a:b, $1 \leq a < b \leq k = O(\log m)$, is $O(\eta |A| m^{-\delta})$, except with probability that does not exceed $\exp(-m^{\delta})$.

Proof. Let δ_3 be such that $\delta_1 < \delta_3 < \delta_2$. Consider the pull-back process for a certain interval T of the partition. Let S_b denote as before the b-th multiples of points of T. Then S_b is a subset of T_bT. Let x denote the number of points of A in S_b. We distinguish two cases.

Case 1: $x > m^{1/4+\delta_3}$. We will apply the pull-back argument to only T_bT and T_aT. We start by a weak bound on the intersection of A and H in S_b. What is the probability that this intersection will exceed $xm^{-\delta}$ in size, where δ is positive but less than δ_1, $\delta_3-\delta_1$, and $\delta_2-\delta_3$? We use our Theorem 3.2.1. We have

$$\alpha = x/m, \ \beta=\eta,$$
$$1+\varepsilon = m^{-\delta}/\eta \geq m^{1/4}.$$

Then

$$\varphi(\varepsilon)\alpha\beta m \geq [(1+\varepsilon)\log(1+\varepsilon)-\varepsilon]\alpha\beta m \geq [(1/4)\log m \ m^{-\delta}/\eta - (m^{-\delta}/\eta - 1)]x\eta$$
$$\geq Cxm^{-\delta} = Cm^{1/4+\delta_3-\delta}$$

for some positive constant C. Thus the probability under consideration is exponentially small, and we may assume that our intersection does not exceed $xm^{-\delta}$. We now pull back this intersection to T_aT, thereby defining S_a. This is a disjoint set from S_b, and applying once more our intersection theorem with ε this time small, say $\varepsilon=1/2$, we immediately conclude that, except with probability not exceeding

$$\exp(-C'\eta xm^{-\delta}) \leq \exp(-m^{\delta}),$$

the number of pairs (u,v) with $u\in H\cap T_aT$, $v\in H\cap A\cap T_bT$, u,v in the ratio a:b, is no greater than

$$O(\eta xm^{-\delta}).$$

Case 2: $x \leq m^{1/4+\delta_3}$. We now simply don't bother with the T_bT step of the above argument. Just pull back the entire $A\cap S_b$ to T_aT in order to obtain S_a. Thus S_a has size $\leq m^{1/4+\delta_3}$, and since we are interested in maximizing the number of pairs (u,v), we will in fact assume that S_a has size $m^{1/4+\delta_3}$. Applying Theorem 3.2.1. to S_a and H we obtain that, except with probability not exceeding

$$\exp(-C'x\eta) \leq \exp(-C'm^{\delta_3-\delta_1}) \leq \exp(-m^{\delta}),$$

the total of pairs (u,v) with $u\in H\cap T_aT$, $v\in A\cap T_bT$ (and *a fortiori* those for which $v\in A\cap H\cap T_aT$) and u,v in the ratio a:b, is no greater than

$$O(x\eta) = O(m^{\delta_3-\delta_1}).$$

Now we sum the contributions over all T. The contributions from Case 1 are $O(\eta(\Sigma_T x)m^{-\delta})$. Since the mapping z → bz is 1-1, we must have $\Sigma_T x \leq |A|$. Thus the total from case 1 is $O(\eta|A|m^{-\delta})$, which is at least $Lm^{\delta_2-\delta_1-\delta}$, by our assumption about the size of A. By taking L sufficiently large (say L = Cklog m, for some constant C) we can make the contribution of the fixpoint intervals negligible compared to this, say no more than $m^{\delta_3-\delta_1}$. Also the contributions of Case 2 are no more than $O(k^3(\log m)m^{\delta_3-\delta_1})$ (just let all T's have a Case 2 contribution), and that too is negligible.

By choosing the δ in the statement of the theorem slightly smaller than the δ we have used in the proof, the result follows.∎

Clearly the results of the last Lemma and Theorem can also be generalized so that they apply to all points and all ratios and types up to some maximum length $k_0 = O(\log m)$ simultaneously.

In a certain light what we have shown is that the occurrence of one arithmetic progression of length k in H influences very little the occurrence of another such progression. These progressions are nearly independent in the sense that they give rise to a distribution analogous to that of independent Bernoulli trials. This is why the results of this section could not have been obtained by variance arguments alone. It is interesting that for k=3 a similar result can be proved using the exponential sums technique of analytic number theory (see Appendix). Unfortunately that proof does not appear to generalize to k>3.

3.5. Double Hashing

A common technique for collision resolution in hashing is known by the name of double hashing. We described the technique briefly in Section 1.2. Here we review some of the definitions, before we embark on the analysis of this technique in the following sections. Our hash table consists of records, each uniquely identified by a key K. Entries in the table are either occupied or empty. The letters h,g will denote hash-functions, i.e., mappings from the set of all possible keys to an appropriate subset of the integers {0,1,...,m-1}, where m is the table size. We will ignore the complication of detecting a full table. With these assumptions the double hashing algorithm is (as in [Knuth2]):

DOUBLE HASHING: Assume m is a *prime* integer. Let h have range $\{0,1,\ldots,m-1\}$ and g have range $\{1,2,\ldots,m-1\}$.

D1. [First hash.] Set $i \leftarrow h(K)$.

D2. [First probe.] If TABLE[i] is empty, go to D6. Otherwise if KEY[i] = K, the algorithm terminates successfully.

D3. [Second hash.] Set $c \leftarrow g(K)$.

D4. [Advance to next.] Set $i \leftarrow i-c$; if now $i<0$, set $i \leftarrow i+m$.

D5. [Compare.] If TABLE[i] is empty, go to D6. Otherwise if KEY[i] = K, the algorithm terminates successfully. Otherwise go back to D4.

D6. [Insert.] Insert new record at TABLE[i].

The primality of m is essential in order to ensure that in step D4 all table positions are generated before a repetition occurs.

A way to measure the performance of a hashing algorithm is by the average number of probes required to insert a new element into the table. Of course this average depends on how full the table is. If the table contains n elements, then this average is denoted by C'_n. Since we speak of averages we need to define the probability distribution involved. The assumption we make, which is both natural from a theoretical viewpoint and quite justified in practice, is that h,g both select each of their allowed values with equal probability, independently of their values on other keys.

If we let m get large, with $n = \alpha m$, α a fixed constant <1, it has been known from simulations that

$$C'_n \sim 1/(1-\alpha)$$

with agreement to 1 or 2 tenths of a percent even for $m \sim 1000$ (see [Bell-Kam], [Brent]). In the following sections we will prove that

$$C'_{\alpha m} = 1/(1-\alpha) + o(1) \quad \text{as } m \rightarrow \infty,$$

provided $\alpha \leq \alpha_0$, where α_0 is some absolute constant (whose value lies between 1/4 and 1/3). Thus we see that for $0 < \alpha \leq \alpha_0$ double hashing is asymptotically equivalent to uniform hashing, a technique we described and analyzed in Section 1.5.

This equivalence is somewhat surprising, since we would expect double hashing to do substantially worse than uniform hashing. The reason for this is that all probes in the case of uniform hashing are independent, while this is not so for double hashing. In other words, double hashing exhibits clustering; the probability that two keys will follow the same path is $O(1/m^2)$ not "zero" ($O(1/m!)$) as for uniform hashing. The bad configurations for double hashing are sets of occupied positions containing an excessive number of arithmetic progressions. Such sets will tend to grow into sets with even more arithmetic progressions, as a bit of thought will show. So it is by no means true that all sets of n occupied entries are equally likely under double hashing. The sets with an abnormally high number of arithmetic progressions are those that will make C'_n large and are also exactly those most likely to be obtained by double hashing. The effect of our results is to show that the clustering effect is negligible in the limit.

We will use the terms *entry*, *cell*, *slot*, and *point* interchangeably to denote a position of the table. The word *element* or the adjective *occupied* will be used to distinguish the occupied positions. If we consider an arithmetic progression x, x+d, x+2d, ..., x+kd (where we interpret all algebraic operations mod m), then d will be called its *distance* and k its *length*. We will speak of it as an arithmetic progression *coming to* x. If x+d, x+2d,...,x+kd all lie in some set S, then we will speak of it as an arithmetic progression *from* S. Given a point x and a set $S \subseteq \{0,1,\ldots,m-1\}$ of cardinality αm, the expected number of arithmetic progressions of length k coming to x from S is approximately $\alpha^k m$, when we consider all such sets equally likely. This is so because there are (m-1) choices for the distance d and for each such progression we have a probability of

$$C(m-k,\alpha m-k)/C(m,\alpha m) \sim \alpha^k$$

of belonging to S. In Section 3.5 we saw that except for a fraction of selections of S which is exponentially small, the number of arithmetic progressions of length k coming from S to x will be in the range $\alpha^k m(1 \pm \delta)$, for any small postive δ. This result gives us hope to prove what we want, as it shows that sets with an abnormally high number of arithmetic progressions are exceedingly rare.

3.6. The Seed Set and the Final Argument.

In this section we prove that double hashing is asymptotically equivalent to uniform hashing for $0<\alpha\leq\alpha_0$ by using the results of the following two sections. Here α_0 denotes an absolute constant, $1/4<\alpha_0<1/3$. Let us prestate here Corollary 3.8.1, which is the result we will need:

COROLLARY 3.8.1. Given any α,β such that $0<\beta<\alpha\leq\alpha_0$, there exists a constant C_α and an initial configuration of βm occupied positions such that for any small positive constant θ, if we add $(\alpha-\beta)$m points to the table using the double hashing process then, except with probability less than $\exp(-C_\theta m^{\delta_0})$, we will arrive at a configuration of αm occupied elements such that for each point x of the table and for each length k, $2\leq k\leq k'_\alpha = C_\alpha \log m$ the number of arithmetic progressions of length k coming to x from the occupied points is $\alpha^k m(1\pm\underline{\theta}_{\alpha,k})$. Here the $\underline{\theta}_{\alpha,k}$ denote relative errors satisfying

(a) $$\sum_{k=2}^{k'_\alpha} \underline{\theta}_{\alpha,k}\alpha^k \leq \theta,$$

and

(b) $$\alpha^{k'_\alpha}(1+\underline{\theta}_{\alpha,k'_\alpha})m \leq m^{1/2-\delta'},$$

where δ' and δ_0 are small positive constnats, and C_θ is a constant depending on θ only.

What is the average number of comparisons we need in order to find an empty slot in the resulting configuration, using double hashing? (Recall that, as in Chapter 1, we count the final probe into an empty slot as a comparison). As we make such a search, let p_l denote the conditional probability that we will make at least $(l+1)$ comparisons before we find an empty slot, $0\leq l<m$, given that we hit at least one of the occupied positions. Thus we must on the first probe (h(K)) select one position among the set S of occupied positions, and then select a distance (g(K)) that leads to an arithmetic progression of length (at least) l among elements of S. The average number of comparisons for an unsuccessful search will then be

(i) $$1 \quad + \quad \alpha \sum_{l=0}^{m-1} \quad p_l.$$

We first dispose of the arithmetic progressions of length greater than k'_α. Let us consider an occupied point $x \in S$ (all such points are equivalent for the computation below). We claim that there are no arithmetic progressions coming from S to x of length exceeding $(m^{1/2-\delta'}+1)k'_\alpha$. This is so, since if we had such a progression, then we would have more than $m^{1/2-\delta'}$ progressions of length k'_α coming to x from S. (If d is the distance of the original progression, then d, 2d, 3d, ..., $(m^{1/2-\delta'}+1)d$ would all be distances of progressions of length k'_α that are subsets of the long progression). But this contradicts condition (b) of the above corollary. Now for lengths between k'_α and $k'_\alpha(m^{1/2-\delta'}+1)$, we can have at most as many arithmetic progressions as we have at k'_α. The total contribution of these to (i) is

$$\underset{\nwarrow \atop \text{choices for } x}{\alpha m} \quad \sum_{k=k'_\alpha}^{k'_\alpha(m^{1/2-\delta'}+1)} \underbrace{m^{1/2-\delta'}}_{\substack{\text{contribution of}\\ \text{all distances}\\ \text{in question}}} \times \quad \underset{\nearrow \atop \substack{\text{probability of choosing}\\ \text{a specific x and a}\\ \text{specific distance}}}{1/m(m-1)}$$

$$= \quad O(k'_\alpha m^{-2\delta'}) = o(1) \text{ as } m \to \infty, \quad \text{since } k'_\alpha = O(\log m).$$

Here we have ignored the contribution of the excluded configurations, but these can contribute at most a total of

$$m \exp(-C_\theta m^{\delta_0}) = o(1) \quad \text{as } m \to \infty$$

maximum number of probes for any search to the mean, and so from now on they will be ignored for good. For the shorter k we see that our corollary implies

$$p_k = \alpha^k(1 \pm \underline{\theta}_{\alpha,k}) \qquad 2 \le k \le k'_\alpha,$$

and certainly $p_0 = 1$, $p_1 = \alpha$. So the contribution of short lengths to Σp_k is

$$1 + \alpha + \sum_{l=2}^{k'_\alpha} \alpha^l(1 \pm \theta_{\alpha,l})$$

$$= \sum_{l=0}^{k'_\alpha} \alpha^l \quad \pm \theta \quad = \quad 1/(1-\alpha) \quad \pm \quad \theta \quad + \quad o(1),$$

where we have used conclusion (a) of the corollary and the fact that

$$\sum_{l=k'_\alpha+1}^{\infty} \alpha^l = o(1) \quad \text{as } m\to\infty$$

since $k'_\alpha \to \infty$ as $m \to \infty$. Combining all of our conclusions, we have proved that the average number of comparisons needed to find an empty position in a table filled up to load factor α as described in the corollary is

$$1 + \alpha/(1-\alpha) \pm \alpha\theta + o(1) =$$

$$\text{(ii)} \qquad 1/(1-\alpha) \pm \alpha\theta + o(1).$$

Unfortunately we are not done, *because we did not start from an empty table.* The double hashing algorithm was applied only after an initial seed of βm points was already strategically placed in the table.

In order to complete our argument, we need to investigate the effect of these initial βm points. We have added $(\alpha-\beta)$m keys using the double hashing process. What if we had added these same $(\alpha-\beta)$m keys to an initially empty table using double hashing? Let us select a specific hash sequence $(h(K_1),g(K_1))$, $(h(K_2)),g(K_2))$,... and so on. Let S denote the set obtained by adding points with this sequence to the initial βm set, and S' the corresponding set obtained by adding points using the same hash sequence to an initially empty table. Then we claim $S'\subseteq S$. Consider the first point K in our sequence, whose insertion would cause an alleged violation of this condition. Either our key K ends up in the same position in both S' and S, in which case there can be no violation, or our key continues on a longer search path in S than it did in S'. But then the location where K ends up in S' must have already been occupied in S, and so again no violation is possible. The above remark implies that the average number of probes to find an empty slot with configuration S is an upper bound for $C'_{(\alpha-\beta)m}$, i.e.,

$$C'_{(\alpha-\beta)m} \le 1/(1-\alpha) + \alpha\theta + o(1),$$

or

$$C'_{\alpha m} \le 1/(1-\alpha-\beta) + (\alpha+\beta)\theta + o(1),$$

by a simple change of variable. (Assume β is so small that $\alpha+\beta<1$.)

Next we get a lower bound for $C'_{\alpha m}$. For this paragraph only O_β will mean O with reference to $\beta \rightarrow 0$. We just saw that if we start with βm points rather than an empty table, we can only do worse. But how much worse? Again let us fix our attention to the particular hash-sequence on hand $(h(K_1),g(K_1))$, $(h(K_2),g(K_2))$... etc. In the set difference S-S' we have βm points. Now suppose we are at the final configuration S and let us look at an arithmetic progression of length k of occupied cells in S coming to x. If this progression contains at least one point in S-S', we shall say that it is destroyed. (This means that it contributed to the the computation of p_l for α but will not contribute to the one for $\alpha-\beta$). No point in S-S' can destroy more than k such progressions, so the total number of progressions of length k coming to x that is destroyed is bounded by $k\beta m$. Of course we can never destroy more arithmetic progressions than there are, which is $\alpha^k m(1+\underline{\theta}_{\alpha,k})$. Now let $k_0 = \log(1/\beta)/\log(1/\alpha)$. Then the number of progressions coming to x of length greater or equal to k_0 that can possibly be destroyed is

$$\sum_{k=k_0}^{m-1} \alpha^k m(1+\underline{\theta}_{\alpha,k}) = O(\alpha^{k_0} m) = O(\beta m),$$

where we estimated the sum as we estimated sum (i) (using also the obvious fact that the errors $\underline{\theta}_{\alpha,k}$ are bounded for fixed k, as $m \rightarrow \infty$). From 1 to k_0 we can destroy at most

$$k_0^2 \ \beta m = O_\beta(\beta \ \log^2(1/\beta) m)$$

arithmetic progressions. Thus the total of destroyed progressions coming to x is

$$O_\beta(\beta \ \log^2(1/\beta) m),$$

and we have shown

$$C'_{\alpha m} \geq 1/(1-\alpha-\beta) - (\alpha+\beta)\theta - O_\beta(\beta \ \log^2(1/\beta)) + o(1)$$

by arguing as in the previous paragraph.

To summarize, we have

$$1/(1-\alpha-\beta) - (\alpha+\beta)\theta - O_\beta(\beta \ \log^2(1/\beta)) + o(1) \leq C'_{\alpha m}$$

$$\leq 1/(1-\alpha-\beta) + (\alpha+\beta)\theta + o(1).$$

Since θ, β can be taken to be arbitrarily small, we have proved.

THEOREM 3.6.1. The average number of comparisons needed to find an empty slot with double hashing in a table of size m, filled up to load factor α, $\alpha \leq \alpha_0$, is

$$C'_{\alpha m} = 1/(1-\alpha) + o(1) \quad \text{as } m \to \infty.$$

3.7. The Lattice Flows and the Extension Process.

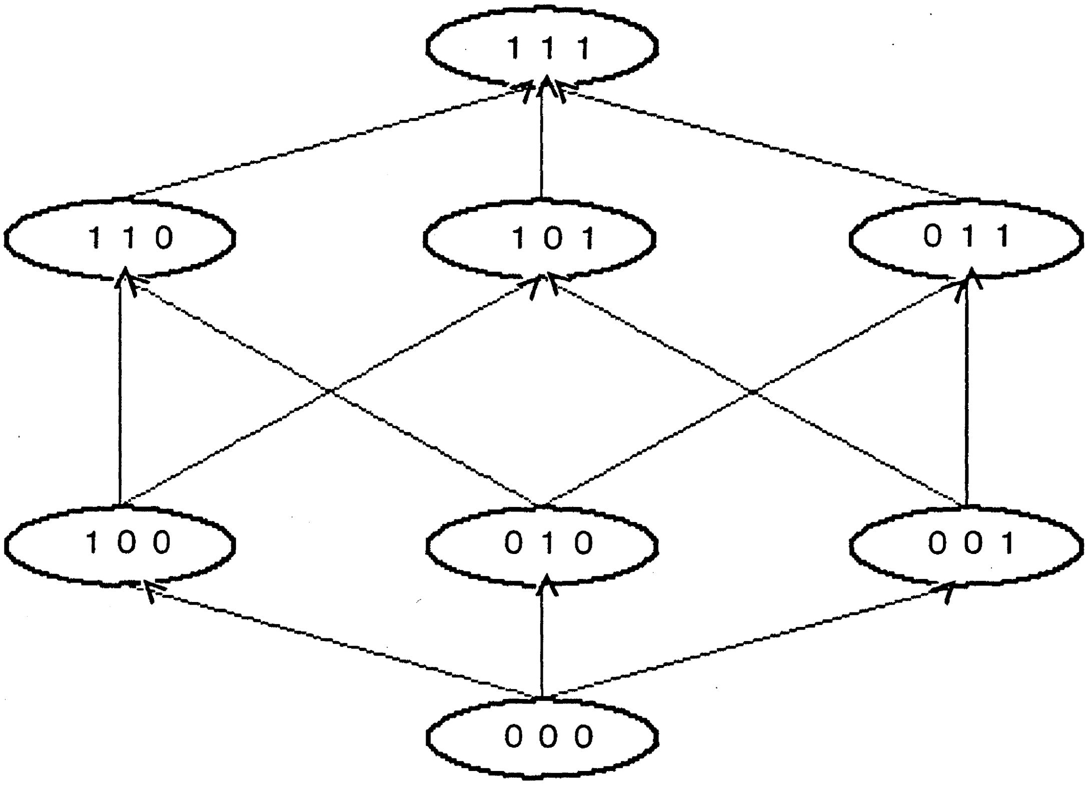
Let x be a point of the table, and let τ be a type of length k with h hits and k-h misses. If γm points of the table are occupied, $0<\gamma<1$, then the expected number of arithmetic progressions of type τ coming to x is $\gamma^h(1-\gamma)^{k-h}m$. In this and the following section we show that if we start with a configuration of βm occupied points in which every point has nearly the expected number of arithmetic progressions of every type and grow this table to αm elements using the double hashing process, then if we only exclude an exponentially small fraction of possibilities, we can be sure that the resulting configuration of αm points will also have nearly the expected number of arithmetic progressions of every type coming to every point.

To illustrate the argument we first discuss how we can prove such a statement if the additional $(\alpha-\beta)m$ elements were randomly inserted. We add the new points in groups of ηm at a time, where η is very small compared to α or β. Suppose we currently have γm elements in the table and are about to add ηm new ones. Fix a point x of the table and consider the arithmetic progressions of length k coming to x. The various types to which these progressions may belong form a boolean lattice, as illustrated by Figure 3.7.1.

As the new ηm points are added, there will be flows upwards in this lattice, that is, some arithmetic progressions will shift into types with more hits. For example, the first point of a progression of type (001) may become occupied by one of the ηm points, whereas the second may stay empty, thus causing the progression to shift into type (101). In order to estimate the magnitude of these inter-type flows we need to introduce some notation.

THE LATTICE OF TYPES OF ARITHMETIC PROGRESSIONS OF A GIVEN LENGTH COMING TO A POINT

k = 3


1 1 1
1 1 0
1 0 1
0 1 1
1 0 0
0 1 0
0 0 1
0 0 0

"1" denotes a hit
"0" denotes a miss

arrows indicate
inter-type flows

Figure 3.7.1.

DEFINITION 3.7.1. Let x be a point of the table, τ a type of length k, i an integer $1 \leq i \leq k$, and Γ a configuration of γm occupied positions; then by $S(i,\tau,x,\Gamma)$ we will denote the set of the i-th points of the arithmetic progressions of type τ coming to x. We also introduce the density

$$\sigma(\tau,x,\Gamma) \quad = \quad |S(i,\tau,x,\Gamma)|/m,$$

which is clearly independent of i (since m is prime).

Throughout the arguments that follow we will be dealing with inequalities on the $\sigma(\tau,x,\Gamma)$. We introduce the symbol $\sigma(\tau,\gamma)$ to stand for any of $\sigma(\tau,x,\Gamma)$, where x ranges over all points and Γ over all non-excluded configurations of γm occupied positions. Thus when we write

$$\sigma(\tau,\gamma) = \lambda(1 \pm \mu),$$

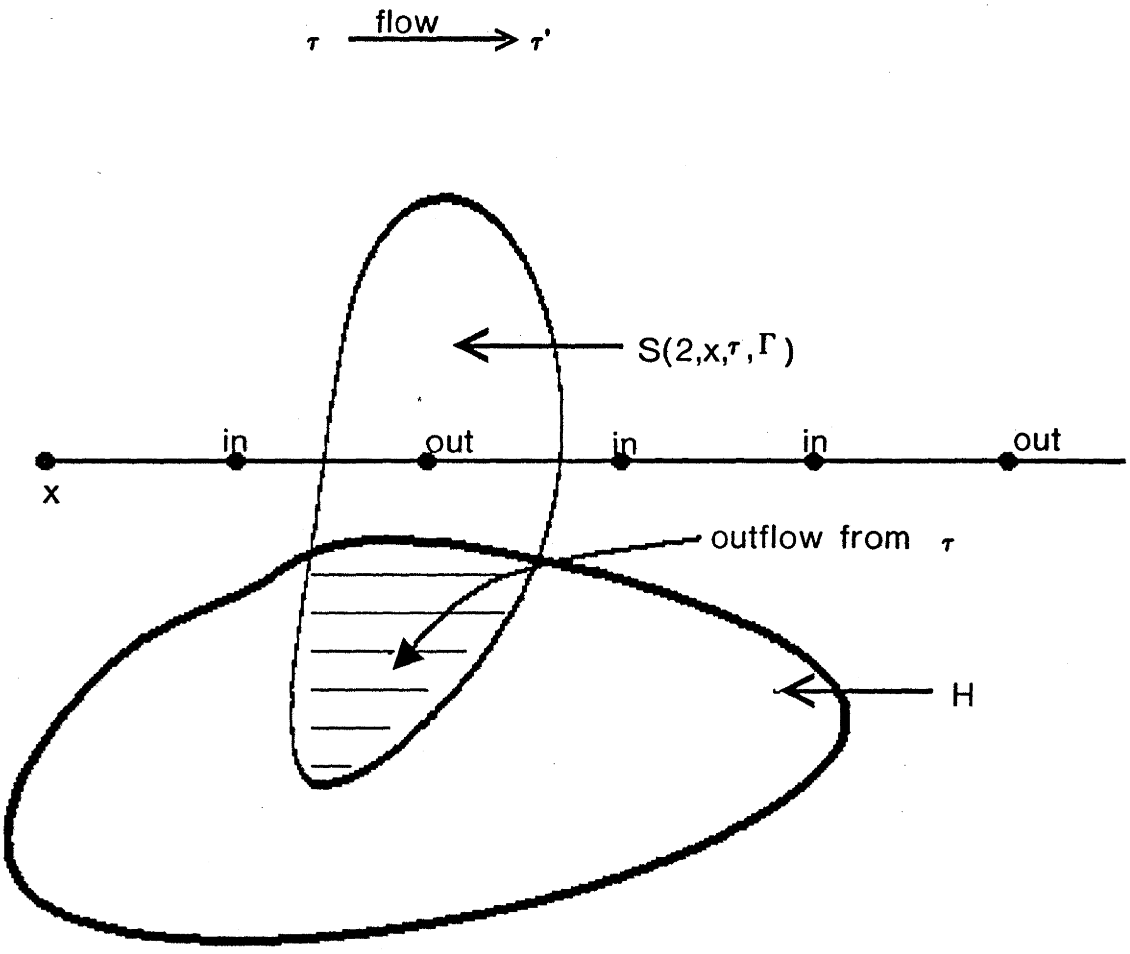
we mean $\lambda(1-\mu) < \sigma(\tau,x,\Gamma) < \lambda/(1-\mu)$ for all x and Γ as described above.

We now present a heuristic argument for the case of random insertions. Assume that our configuration Γ is such that $\sigma(\tau,\gamma) = \gamma^h(1-\gamma)^{k-h}$ for all types τ, where h denotes the number of hits and k-h the number of misses of the type. Thus $S(i,\tau,x,\Gamma) = \gamma^h(1-\gamma)^{k-h}m$. What happens to the arithmetic progressions of type τ coming to x as the new ηm points are added? Consider a type τ' which is τ except a hit of τ in the i-th position is a miss in τ', e.g., $\tau = (101)$, $\tau' = (001)$ in the example above. Then for each point of $S(i,\tau,x,\Gamma)$ that is hit by the ηm, a progression may change from type τ' to type τ. This is illustrated in Figure 3.7.2.

There are $(1-\gamma)m$ unoccupied elements, of which we are choosing ηm, thus the probability of selecting a point is $\eta/(1-\gamma)$. By hypothesis the size of $S(i,\tau,x,\Gamma)$ is $\gamma^{h-1}(1-\gamma)^{k-h+1}m$, and therefore the expected size of its intersection with the ηm is $\eta\gamma^{h-1}(1-\gamma)^{k-h}m$. Since τ has h hits, there are h possible positions at which such inflows into type τ can occur (i.e. there are h possible feeder types τ'), for a total of $h\eta\gamma^{h-1}(1-\gamma)^{k-h}m$. Now some of the progresions of type τ can move out of this type. For each i that corresponds to a miss of τ, this will happen whenever the set $S(i,\tau,x,\Gamma)$ is intersected by the ηm. We easily compute the size of the outflow to be $(k-h)\eta\gamma^h(1-\gamma)^{k-h-1}m$. In the above we have ignored the possibility that a transition between two types can occur with more than one of the points of a progression being hit by the ηm. Any flows arising out of such transitions, however, will have an

Figure 3.7.2.

THE MECHANISM OF INTER-TYPE FLOWS


τ
flow
τ'
S(2,x,τ,Γ)
in
out
in
in
out
x
outflow from τ
H

expected magnitude of $O(k^2\eta^2m)$ and since we take η to be very small, they can be ignored. To total up, when the new ηm points have been added, the expected number of arithmetic progressions of type τ coming to x will be

$$\gamma^h(1-\gamma)^{k-h}m+h\eta\gamma^{h-1}(1-\gamma)^{k-h}m-(k-h)\eta\gamma^h(1-\gamma)^{k-h-1}m$$

which is $(\gamma+\eta)^h(1-\gamma-\eta)^{k-h}m$ if again we ignore $O(k^2\eta^2m)$ terms.

Thus $S(i,\tau,x,\Gamma) = (\gamma+\eta)^h(1-\gamma-\eta)^{k-h}$ on the average, as we had hoped. By iterating this heuristic argument we see how we can grow from βm to αm elements while having the expected number of arithmetic progressions of any type at each point at each step.

In the next section we will show how to carry out this argument rigorously. For the remainder of the current section we confine ourselves to some definitions and general remarks. We shall use the term *the extension process* for this process of building up the table we are describing. This process consists of steps of adding ηm points at a time. During each step, given any two types of progressions coming to a point x, there may be transitions of actual progressions from one type to the other. These inter-type transitions will be called *flows.* For each type we will have a certain *inflow* and *outflow* of progressions from it. Naturally we cannot assume that a type will have exactly the expected number of progressions, as we have done in the heuristic argument above. We introduce relative errors $\theta_{\gamma,\tau}$ on this expected value that describe the deviation we are willing to allow. In other words, when we are at load factor γ, we assume that for each point x and each type τ of length k (k not exceeding a certain maximum) and h hits, we will have

$$\gamma^h(1-\gamma)^{k-h}m(1\pm\theta_{\gamma,k})$$

arithmetic progressions of type τ coming to x. Here we have already adopted the convention that we will follow in the actual argument and suppressed the dependency of $\theta_{\gamma,\tau}$ on anything but the length k of τ. We will find that the errors $\theta_{\gamma,k}$ grow faster for larger k, but if we compute the *total* number of arithmetic progressions coming to x of types consisting entirely of hits, then the relative error on this total we will be able to make as small as we please. Now double hashing chooses each of the empty points with probability proportional to the number of arithmetic progressions coming from the occupied points to that point. The above remark then implies that during the current step every empty position is nearly equally likely to be filled. So we are not too far from the random situation. But how can we be sure that we will maintain the same good situation during the next step?

Here we invoke Theorem 3.2.1 to assure that all intersections between the ηm points and the various sets $S(i,\tau,x,\Gamma)$ of Definition 3.7.1 are nearly the same size. In doing this we exclude an exponentially small fraction of choices of the ηm points, while increasing the relative errors $\theta_{\gamma,k}$ to $\theta_{\gamma+\eta,k}$ for the next step. We will speak of using Theorem 3.2.1 for *controlling the intersections*, and therefore the flows. In order to keep the error propagation equations for $\theta_{\gamma,k}$ relatively clean, we will allow certain additional absolute errors as well (the "residual" progressions of the next section). During any step, if there is a number of progressions flowing between two types that is allowed by our control but cannot be accounted for in the relative errors we allow, this number we will speak of as an *excessive flow*. The gist of the argument then is that by excluding an exponentially small fraction of possibilities, we maintain at each step every empty position nearly equally likely to be filled. We never give clustering a chance to build up a really bad configuration.

We now make a number of remarks that the reader should keep in mind while reading the next section.

Remark 1. The types that ultimately play a role in double hashing are those consisting entirely of hits. Because, however, the population of types changes by inter-type flows, we have to attempt to control all types at once.

Remark 2. Suppose we wish to maximize the number of progressions in a type τ consisting of k hits. During each step the significant inflows into τ are those from types with k-1 hits. Obviously we should maximize these inflows. Now these inflows are also outflows from the "feeder" types one step below in the lattice. In order to maximize those same inflows during the next step, we want to maximize the growth of the feeder types during the current step. But these types have their outflows already chosen, so the best we can do is to maximize the inflows into them. An inductive extension of this argument shows that all flows in the lattice should take their maximum allowed value during every step, if we are interested in maximizing the growth at the apex of the lattice. Similarly if we wish to minimize this growth, all flows should be minimized. The point made here is important and somewhat subtle, and the reader should dwell on it for a moment. Another way to see the point is this. Consider one of the sets S corresponding to one of the feeder types. At the current step a fraction ρ_1 of S will flow, where ρ_1 is allowed to vary within certain limits. At the next step a fraction ρ_2 of the part of S that is left will flow, and so on, say up to ρ_ν. Then it is simple to see that the total fraction of S that has flowed is

$$1 - \prod_{i=1}^{\nu} (1-\rho_i)$$

and this expression is maximized when all of the ρ_i are maximized. The intuitive interpretation of this is that if we wish to maximize the total flow between two types, we should never trade the certainty of a specific transition in the current step for the probability of that same transition in some future step.

Remark 3. If we are interested in maximizing the flows, it will only hurt our upper bound to make any of the sets S of Definition 3.7.1. that partake in the controlled intersections larger than it really is.

Remark 4. Since we are dealing with non-negative quantities, a relative error smaller than -1 clearly does not make sense. We do, however, allow such fictitiously large negative errors in the argument of the next section, since they can only make our lower bounds worse and they avoid consideration of special cases.

Remark 5. If P(m) denotes any polynomial in m, C, δ_1, δ_2 constants with $C>0$, $\delta_2>\delta_1>0$, then for m sufficiently large

$$P(m)\ \exp(-Cm^{\delta_2}) < \exp(-Cm^{\delta_1}).$$

Remark 6. let θ denote an arbitrarily small positive number and let $\psi(m)$ be a quantity which is $o(1)$ as $m \to \infty$. Then we will say that ψ can be incorporated in θ to mean that, given any positive constant θ', for m sufficiently large we can assume that the sum $\theta+\psi(m)$ does not exceed θ'. We use this terminology on a number of occasions. This is justified because it will be trivial to check that *the sum* of the $\psi(m)$ over all instances of the terminology that refer to the same θ is $o(1)$.

Remark 7. We will make some use of the O,o -notations. They always refer to $m \to \infty$, and the implied constants are either absolute or depend at most on α, which is a constant of the entire problem. In Corollary 3.8.1. we also use the notations $\ll$, $\approx$ with their usual heuristic meaning. If the reader wishes to have an exact meaning, then he may take, in the context where these occur, $f \approx g$ to mean $gm^{-\delta} \leq f \leq gm^{\delta}$, and $f \ll g$ to mean $f \leq gm^{-\delta}$, for some small positive δ.

Remark 8. The reader should realize that the process of intertype flows we have described is only a model for what occurs in the real table. The model will be used to bound the number of progressions we can actually have in the table. It need not be the case that the flows we use in the estimations of the next section can actually be realized by some sequence of insertions into the actual table.

3.8. The Propagation of Errors and the Impotence of Clustering

We will now carry out a precise estimation of the error propagation in the extension process. We assume α,β are fixed constants, β small, $0<\beta<\alpha<1$. In the course of the computation we will find that we have to restrict α to be below some absolute constant α_0, $\alpha_0 < 1$. We take

$$\eta = m^{-1/4-\delta_1},$$

and define

$$k_\beta = [(3/4-\delta_2)/\log(1/\beta)] \log m \qquad (\text{so } \beta^{k_\beta} m = m^{1/4+\delta_2})$$

(i)

$$k_\alpha = [(1/2+\delta_3)/\log(1/\alpha)] \log m \qquad (\text{so } \alpha^{k_\alpha} m = m^{1/2-\delta_3})$$

where δ_0, δ_1, δ_2, δ_3, δ_4 are small positive constants such that

(ii) $\delta_2 > \delta_1,\ \delta_2-\delta_1 > \delta_4 > \delta_0 > 0$ (δ_0, δ_4 will be used later).

Our choice for η is a compromise between two conflicting requirements. On the one hand we want to make η as large as possible so as to get the maximum benefit from the law of large numbers and Theorem 3.2.1. On the other hand we want to take η sufficiently small so that we can ignore the interactions of the ηm points among themselves.

During the extension process we need to maintain control over arithmetic progressions of length k_α since the argument of Section 3.6 depends heavily on our ability to push the number of arithmetic progressions of length k_α below some power less than $m^{1/2}$. Unfortunately in the early stages of the extension process we are then out of luck. For types τ of length k_α and many hits, the expected number of progressions of that type coming to a point will be too small to either assert anything initially, or to control the intertype flows by bounding the size of the intersections with the ηm points. To circumvent this shortcoming we introduce a technical device. For each point x and for each type τ of length between k_β and k_α we introduce an initial maximum positive "error" of size $E_0 = m^{1/4+\delta_2}$ in the number of arithmetic progressions of type τ coming to x. This error is in addition to the regular relative errors discussed in Section 3.7. As we will see, it provides us with a way of masking out the fact that we cannot control the size of the relative errors during the early stages of the extension process. These additional progressions will of course flow among the types like the normal ones we have already considered. We will call them the *residual* progressions and will control their flows independently of the regular progressions.

We will ignore the outflow of residual progressions from any given type. Thus their number can only grow and will never become less than E_0. By analogy with Definition 3.7.1. we introduce the notations $R(i,\tau,x,\Gamma)$, $\rho(\tau,x,\Gamma)$ to denote the corresponding quantities for the residual progressions. Thus $\rho(\tau,x,\Gamma) \geq m^{-3/4+\delta_1}$. If at any moment during the extension process we have

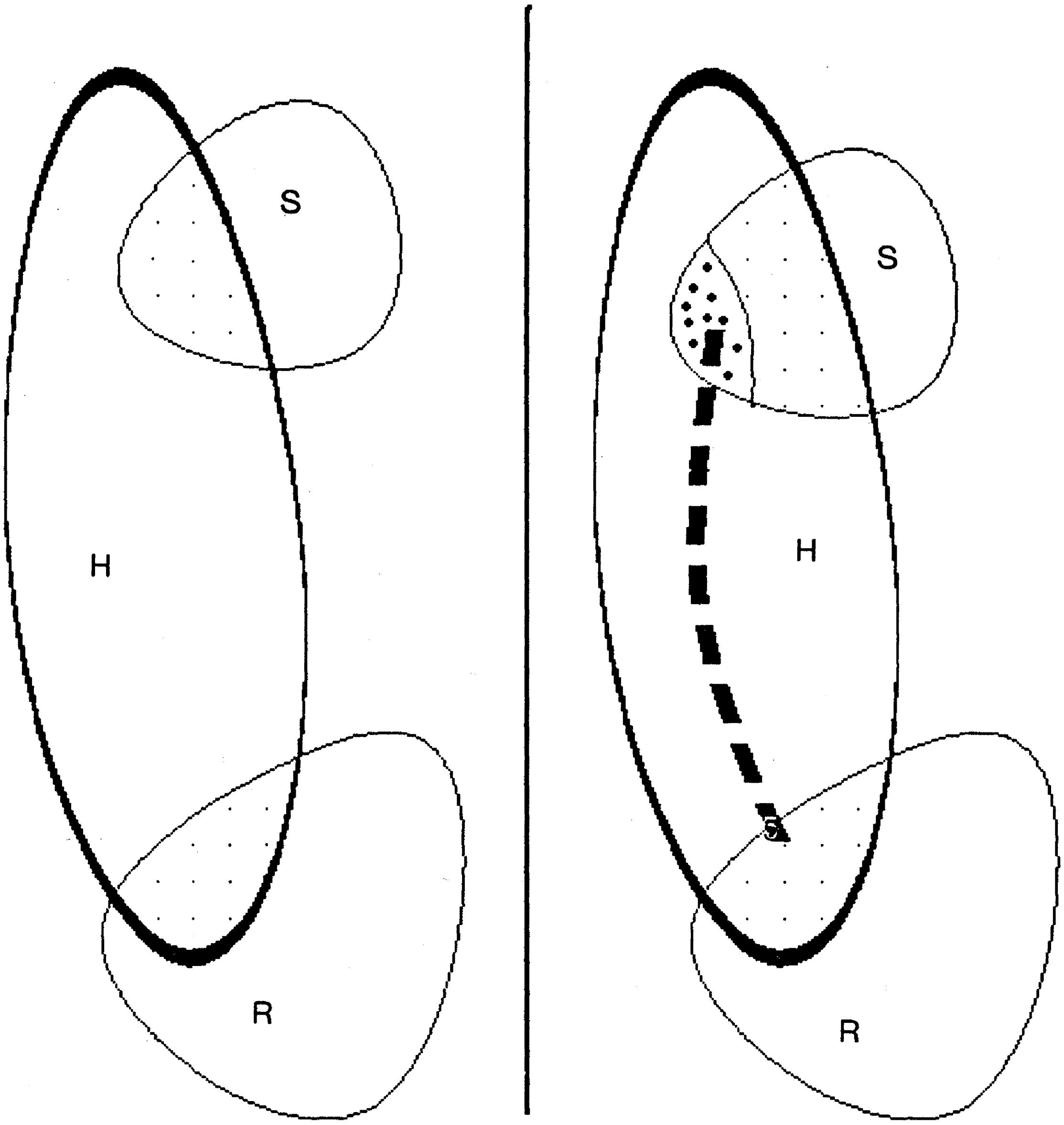
$$\sigma(\tau,x,\Gamma) < \rho(\tau,x,\Gamma)$$

then we will not attempt to control the intersection of any of $S(i,\tau,x,\Gamma)$ with the ηm. Instead we will control *only* $S(i,\tau,x,\Gamma) \cup R(i,\tau,x,\Gamma)$, which has cardinality $(\sigma(\tau,x,\Gamma)+\rho(\tau,x,\Gamma))m$. We also use the notation

$$r(\tau,x,\Gamma) = |R(i,\tau,x,\Gamma)|.$$

If the intersection of the ηm with $S(i,\tau,x,\Gamma)$ was excessively large, then any excess we will relabel as *residual* progressions for the receiving type. Thus we can guarantee that none of the regular flows (i.e., flows of regular progressions) will be excessive, by allowing sometimes the residual flows to be excessively large (by at most the same relative error). The quantitative argument will be given in the proof of Theorem 3.8.1. Figure 3.8.1. attempts to summarize this camouflaging with the residual progressions.

Figure 3.8.1. CAMOUFLAGING WITH THE RESIDUAL PROGRESSIONS



Case 1: $|R| \leq |S|$ Case 2: $|R| > |S|$

DEFINITION 3.8.1. The generic variable ρ_γ wll stand for any of $\rho(\tau,x,\Gamma)$, the densities of the residual progressions.

As we saw in the last section, it is our goal to perform the extension process so that at each step all empty points are nearly equally likely to be filled. Since at each step we introduce not one but ηm points all at once, we have to understand the interactions among the ηm points themselves. It is possible that an initial fragment of the ηm points will be placed so badly that it will greatly affect where the remaining of the ηm points will go. This, however, can only occur if during an insertion one of the ηm points interacts heavily with those previously inserted.

DEFINITION 3.8.2. Suppose we have a configuration Γ of γm occupied positions and are inserting ηm additional points. An insertion of one of these points will be called *bad* if its probe path (i.e., the sequence of examined points before insertion)

(1) contains an initial segment of length at least k_α consisting of positions of the γm and at most one position occupied by one of the ηm points,
or
(2) contains (at least) two of the m points among its first k_α (or fewer) steps.

An insertion which is not bad will be called *good.* We let b_γ denote the total number of bad insertions that have occurred when we reach a load factor of γ.

Figure 3.8.2. illustrates the different cases of good and bad insertions.

What we will prove below is that the conditional probabilities that any two empty positions will be filled, given that they are filled with good insertions, are nearly equal. We introduce the quantity χ_γ to capture the relative error in the probabilities (recall $\theta_{\gamma,1}$=0).

DEFINITION 3.8.3. We let

$$\chi_\gamma = \sum_{k=2}^{k_\alpha} \gamma^k \theta_{\gamma,k}.$$

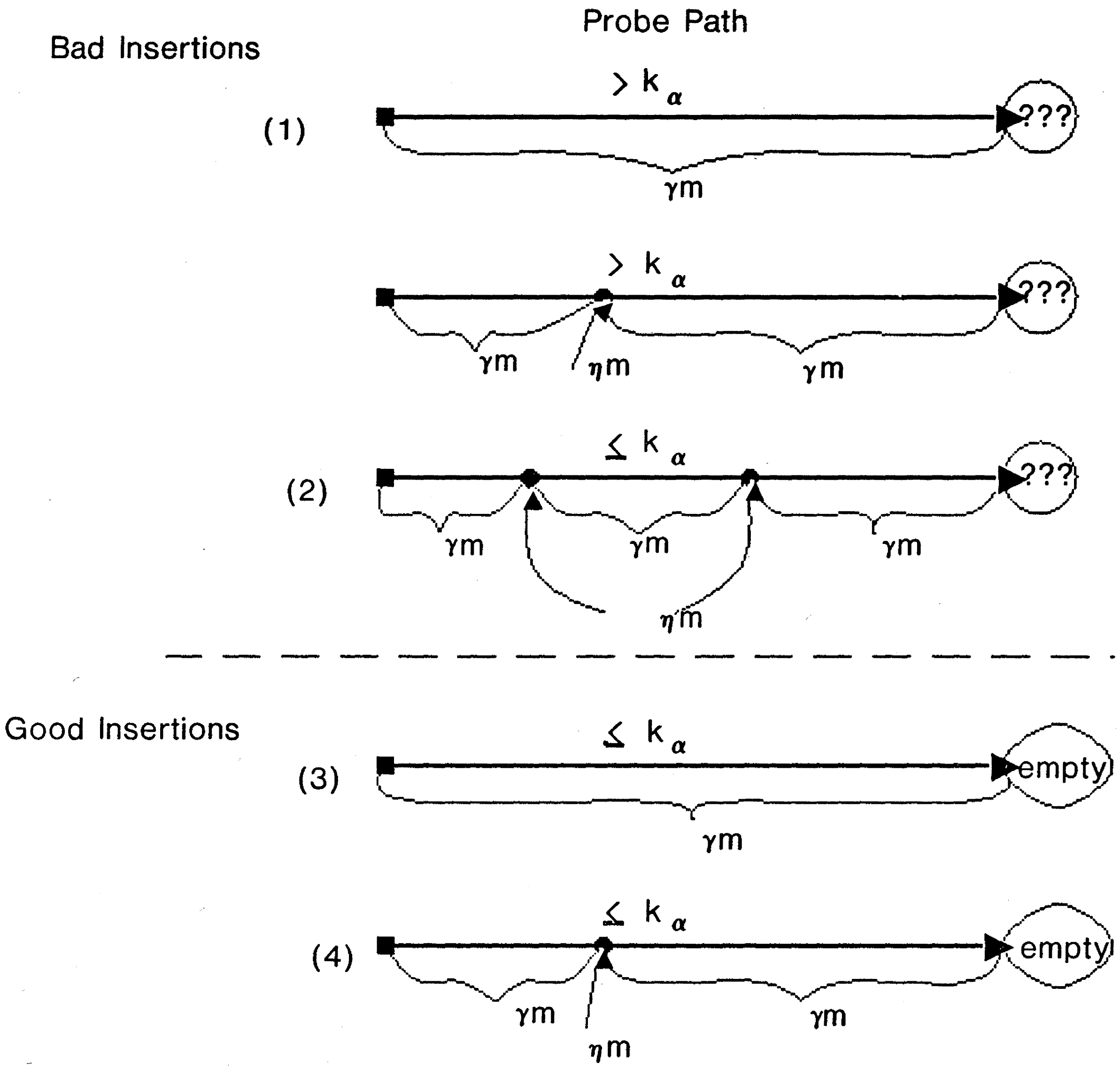

Probe Path
Bad Insertions
(1)
> k α
???
γm
> k α
???
γm
ηm
γm
(2)
≤ k α
???
γm
γm
γm
ηm
Good Insertions
(3)
≤ k α
empty
γm
(4)
≤ k α
empty
γm
γm
ηm

Figure 3.8.2 THE GOOD AND BAD INSERTIONS

We now have all the concepts we need to begin the quantitative argument.

THEOREM 3.8.1. Let α, β be positive constants such that $0<\beta<\alpha<\alpha_0$. There exist absolute positive constants s, D such that, given an arbitrarily small positive constant θ, there exist positive constants i_θ, C_θ (tending to 0 as $\theta \rightarrow 0$) such that: if we begin the extension process with a configuration of βm elements placed so that for each point x and type τ of length less than or equal to k_α we have the expected number of arithmetic progressions of that type coming to x within a relative error of i_θ and (for those τ of length at least k_β) a residual error of at most $E_0 = m^{1/4+\delta_2}$ progressions, then, except with probaiblity $\exp(-C_\theta m^{\delta_0})$ where δ_0 is a constant, $0<\delta_0<\delta_2-\delta_1$ δ_1, δ_2 as defined by η, k_α in (i), when we reach a load factor γ we will have

(a) $\theta_{\gamma,k} \leq (1+i_\theta)e^{5\theta(\gamma^s/s)} - 1$ for $2 \leq k \leq k_\alpha$

(b) $\chi_\gamma \leq \theta\gamma^s$

(c) $\rho_\gamma m \leq E_0 m^{(1/2 + \delta_3)\log[1 + 2\log(1/(1-\gamma))]/\log(1/\alpha)}$

(d) $b_\gamma \leq D_\gamma m^{1/2-\delta}$, with δ, $0<\delta<\delta_3$, $2\delta_1$, a constant,

where θ_γ, k, χ_γ, ρ_γ, b_γ are as given by Definitions 3.7.2, 3.8.3, 3.8.1, and 3.8.2, respectively.

Proof. We will prove assertions (a), (b), (c), and (d) by induction on the number of steps in the extension process. Thus we will assume that they hold for γ and prove them for $\gamma+\eta$. For $\gamma=\beta$ all assertions are true trivially, except for (b) that requires that we take $i_\theta \leq \theta_\beta \beta^s(1-\beta)$, as we certainly can. We will see how to choose the constants D, s in the course of the proof.

The proof is in two parts. First we examine the effect of the bad insertions, and second we look at the propagation of the errors.

What is the probability of a bad insertion? Let us go back to Definition 3.8.2. An initial segment of length at least k_α will be entirely within the γm with probability $\sigma(\tau_0,\gamma)$, where τ_0

is the type of k_α hits. Similarly, the probability of encountering one of the ηm points in this segment is certainly bounded by $k_\alpha \sigma(\tau_1, \gamma)$, where τ_1 denotes any type of length k_α and $k_\alpha - 1$ hits. Thus the probability of condition (1) of the definition being satisfied does not exceed

$$\text{(iii)} \qquad \gamma^{k_\alpha}(1+\theta_{\gamma,k_\alpha}) + k_\alpha \gamma^{k_\alpha - 1}(1-\gamma)(1+\theta_{\gamma,k_\alpha}).$$

We estimate the probability that condition (2) will be satisfied somewhat differently. We ask how many pairs (h(k), g(k)) are there that lead to a probe path satisfying (2). The probe path is completely specified once we know the two ηm poitns involved, and the positions of the two points on the path, say they are the b-th and c-th points respectively. Since we can take $1 \leq b < c \leq k_\alpha$ we have at most ½ $k_\alpha^2 \, \eta^2 \, m^2$ distinct probe paths. Each candidate pair (h(k),g(k)) defines a distinct path. Since each pair occurs with probability 1/m(m-1), we have an overall probability (per insertion) of satisfying (2) that is bounded by $k_\alpha^2 \eta^2$.

From assertion (a) we have

$$\theta_{\gamma,k_\alpha} \leq (1+i_\theta)e^{5\theta(\gamma^s/s)k_\alpha - 1}$$

and since

$$k_\alpha = [(1/2+\delta_3) \log m]/[\log 1/\alpha]$$

it follows that

$$\theta_{\gamma,k_\alpha} \leq (1+i_\theta)m^{5\theta(\gamma^s/s)[[(1/2)+\delta_3]/(\log 1/\alpha)]}$$

which can be made $\leq m^{\delta_5}$ by taking θ sufficiently small, where δ_5 is such that $0 < \delta_5 < \delta_3 - \delta$, $2\delta_1 - \delta$. Thus the quantity specified in (iii) is less than or equal to $m^{-1/2-\delta_6}$ for some $\delta_6 > \delta$, and so is $k_\alpha^2 \eta^2$ as the reader can easily check. The residual progressions of the types accounted for in (iii) have to be added in also, of course, but their number as given by (c) is less than or equal to

$$\frac{m^{1/4+\delta_2} \quad m^{(1/2+\delta_3)\log[1+3\log(1/(1-\gamma))]/\log(1/\alpha)}}{m}$$

which is less than $m^{-1/2-\delta_6}$ for $\gamma \leq \alpha \leq \alpha_0$ as can be easily checked. We will encounter this α_0 later also, so we will not dwell on its value any longer here. Thus we have proved

Claim 1. The probability of a bad insertion is never greater than $m^{-1/2-\delta}$ for $\beta \leq \gamma \leq \alpha \leq \alpha_0$.

By Theorem 3.2.1 (or its equivalent for Bernoulli trials) the probability that we will have more than $Dm^{-1/2-\delta}\eta m = D\eta m^{1/2-\delta}$ bad insertions, for some constant D slightly larger than 1, is less than $\exp(-C(D-1)^2\eta m^{1/2-\delta}) < \exp(-m^{1/4-2\delta-\delta_1})$ and thus this event can be excluded. Therefore we can assert that at load factor $\gamma+\eta$ the total of bad insertions will not exceed

$$D(\gamma+\eta)m^{1/2-\delta},$$

as we desire in order to prove (d). Although we cannot say anything about where the badly inserted points will go, their number is so small that, as we shall see, they cannot destroy the final assertion of regularity of our configuration.

We next show that any two empty positions have nearly equal probabilities of being filled with good insertions. Under double hashing the probability that a given empty position will be filled is proportional to the number of arithmetic progressions coming to that position from the occupied positions. Recall also that in a good insertion, the probe path is at most k_α long and in this path at most one of the new ηm points can occur. Let x be any empty point. The number of regular (i.e. non-residual) arithmetic progressions of length k, $0 \leq k \leq k_\alpha$, coming to x from the occupied points is by assumption $\gamma^k(1\pm\theta_{\gamma,k})m$, for a total of

$$\sum_{k=0}^{k_\alpha} \gamma^k(1\pm\theta_{\gamma,k})m \ ,$$

or the discrepancy over the expected value is in absolute value at most

$$\left(\sum_{k=0}^{k_\alpha} \gamma^k\theta_{\gamma,k}\right) m = \chi_\gamma m \ .$$

Each of the new ηm points can occur in the path, and each such point can introduce at most k additional progressions of length k coming to x (by being the 1st, 2nd,...,k-th point of the progression), for a total of

$$\text{(iv)} \qquad \eta m \sum_{k=0}^{k_\alpha} k \quad \leq \quad k_\alpha^2 \eta m.$$

The number of residual progressions coming to x is at most

$$\text{(v)} \qquad \sum_{k=k_\beta}^{k_\alpha} \rho_\gamma m \quad \leq \quad k_\alpha m^{1/2-\delta}$$

for $\alpha \leq \alpha_0$. Finally the previously badly inserted points can introduce each at most k progressions of length k, for a total as in (iv) of at most

$$\text{(vi)} \qquad b_\gamma \sum_{k=0}^{k_\alpha} k \quad \leq \quad k_\alpha{}^2 D\gamma m^{1/2-\delta}$$

progressions. We only demand in (b) that χ_γ can be made as small as any prescribed constant, and so the combined effect of (iv), (v), and (vi) can be accounted for by asserting that the deviation of the number of the arithmetic progressions coming to x from the expected value does not exceed $2\chi_\gamma m$. Thus we have proved

Claim 2. The probability that at a certain moment any specified empty point will be filled with a good insertion during the γ to $\gamma+\eta$ step is $((1\pm2\chi_\gamma)/(1-\gamma))m$, *independently* of where any previously inserted elements among the ηm were located.

(We have written $2\chi_\gamma$ instead of $2(1-\gamma)\chi_\gamma$ so as to incorporate the error that the $(1-\gamma)$ in the denominator can really vary between $(1-\gamma)$ and $(1-\gamma-\eta)$.) The unavoidable bad insertions and the above small deviation from randomness is the way that clustering manifests itself in this argument. When we insert the new ηm points, the probability that an empty position will be filled is

$$\text{(vii)} \qquad \eta^* = \eta(1\pm2\chi_\gamma)/(1-\gamma).$$

This ignores the effect of the bad insertions, but their contribution can easily be incorporated in the over-generous factor of 2 introduced above, since their number is $O(m^{1/2-\delta})$ which is much less than $\eta m = m^{3/4-\delta_1}$.

We are now ready to begin excluding those choices of the ηm points that would cause any of the inter-type lattice flows to be excessively different from the expected value. We do this simultaneously for the lattices corresponding to all k, $2 \le k \le k_\alpha$ (k=0,1 cannot vary from the average) and all points x. We control the flows by allowing a maximum relative error of θ_0 for the intersections of our ηm points with each of the sets $S(i,\tau,x,\Gamma)$ of Definition 3.7.1, where Γ denotes our current configuration of γm occupied positions. By Theorem 3.2.1 we can do this while excluding only an exponentially small fraction of the choices of the ηm points as long as the expected size of the intersection is not too small. At the beginning of this section we introduced the residual progressions as a device for handling the small $S(i,\tau,x,\Gamma)$. For each i, τ, and x we demand that the intersections of both $S(i,\tau,x,\Gamma)$ and $R(i,\tau,x,\Gamma)$ with the ηm are within $(1\pm\theta_0)$ of what we expect *if* $|S(i,\tau,x,\Gamma)| \ge |R(i,\tau,x,\Gamma)|$, otherwise we only demand this of the (disjoint) union $S(i,\tau,x,\Gamma) \cup R(i,\tau,x,\Gamma)$. In the latter case the intersection will have up to $(\sigma(\tau,x,\Gamma)+\rho(\tau,x,\Gamma))\ \eta^*(1+\theta_0)m$ points. By relabelling some regular progressions as residual we can then still claim that the flow corresponding to the intersection of $S(i,\tau,x,\Gamma)$ with the ηm is $\eta^*\ \sigma(\tau,x,\Gamma)\ (1\pm\theta_0)m$, provided we allow the flow corresponding to $R(i,\tau,x,\Gamma)$ to get as large as $\rho(\tau,x,\Gamma)\ \eta^*(1+\theta_0)m$. Furthermore now no set whose intersection with the m we desire to control has cardinality less than $E_0=m^{1/4+\delta_2}$. Theorem 3.2.1 then implies

Claim 3. During the step from load factor γ to load factor $\gamma+\eta$, if we exclude a fraction of choices of the ηm points that does not exceed $\exp(-C_\theta m^{\delta_4})$ (for δ_4 as in (ii)), then we can assume that the intersection of the ηm points with each of $S(i,\tau,x,\Gamma)$ (τ a type of length at most k_α) will have cardinality $\sigma(\tau,x,\Gamma)\ \eta^*(1\pm\theta_0)m$, and the intersection of the ηm with each of $R(i,\tau,x,\Gamma)$ will not be larger than $\rho(\tau,x,\Gamma)\eta^*(1+\theta_0)m$.

We now compute the relative error $\theta_{\gamma+\eta,k}$ in terms of $\theta_{\gamma,k}$. Let τ, the type we are now considering, have length k and *l* hits. We saw in Section 3.7. that in order to maximize the relative error for the type of k hits, we may assume that all inter-type flows are maximal. As we will see momentarily, we can ignore any flows caused by progressions hit by more than one of the ηm points. Thus the maximal number of progressions of type τ we can have at any point when we reach load factor $(\gamma+\eta)$ is

$$\Big[\underbrace{\gamma^{l}(1-\gamma)^{k-l}(1+\theta_{\gamma,k})}_{\text{already there}}+\underbrace{l\gamma^{l-1}(1-\gamma)^{k-l+1}(1+\theta_{\gamma,k})\eta^*(1+\theta_0)}_{\text{inflow}}\ \underbrace{-(k-l)\gamma^{l}(1-\gamma)^{k-l}(1\underset{\text{"+" since \textit{all} flows are maximal}}{+}\theta_{\gamma,k})\eta^*(1+\theta_0)}_{\text{outflow}}\Big]m$$

Figure 3.8.3. illustrates the inflow and outflow of progressions from a type.

Ignoring the factor of m we can write the above expression as

$$\gamma^{l}(1-\gamma)^{k-l}(1+\theta_{\gamma,k})+l\gamma^{l-1}(1-\gamma)^{k-l+1}(1+\theta_{\gamma,k})(1+\theta_0)\eta(1+2\chi_\gamma)/(1-\gamma)$$
$$-(k-l)\gamma^{l}(1-\gamma)^{k-l}(1+\theta_{\gamma,k})(1+\theta_0)\eta(1+2\chi_\gamma)/(1-\gamma).$$

This has to equal $(\gamma+\eta)^{l}(1-\gamma-\eta)^{k-l}(1+\theta_{\gamma+\eta,k})$, and so we get

$$1 + \theta_{\gamma+\eta,k} = \frac{\gamma^{l}(1-\gamma)^{k-l}}{(\gamma+\eta)^{l}(1-\gamma-\eta)^{k-l}}\Big[1 + \theta_{\gamma,k} + [l\eta/\gamma](1+\theta_{\gamma,k})(1+2\chi_\gamma)(1+\theta_0)$$
$$-[(k-l)\eta/(1-\gamma)](1+\theta_{\gamma,k})(1+2\chi_\gamma)(1+\theta_0)\Big].$$

Now

$$\frac{\gamma^{l}(1-\gamma)^{k-l}}{(\gamma+\eta)^{l}(1-\gamma-\eta)^{k-l}} = 1 - l\eta/\gamma + (k-l)\eta/(1-\gamma) + O(\eta^2).$$

If we ignore the η^2 terms, then we can rewrite the above as

$$\theta_{\gamma+\eta,k} = \theta_{\gamma,k} + (\eta l/\gamma)(\theta_0+2\chi_\gamma+2\theta_0\chi_\gamma)\theta_{\gamma,k}$$

$$+ (\eta l/\gamma)(\theta_0+2\chi_\gamma+2\theta_0\chi_\gamma)$$

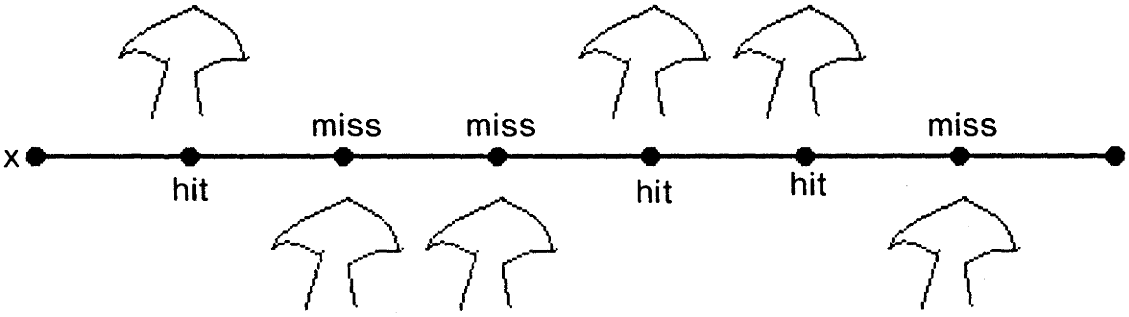
$$- [\eta(k-l)/(1-\gamma)]\ (\theta_0+2\chi_\gamma+2\theta_0\chi_\gamma)\theta_{\gamma,k}$$

$$- [\eta(k-l)/(1-\gamma)]\ (\theta_0+2\chi_\gamma+2\theta_0\chi_\gamma).$$

The effect of the η^2 terms can be incorporated in the constants θ_0 or χ_γ, and so these terms can be justifiably ignored. We maximize the error $\theta_{\gamma+\eta,k}$ by taking l=k above, so our final error propagation equation becomes

(ix) $\theta_{\gamma+\eta,k} = \theta_{\gamma,k} + (\eta k/\gamma)(\theta_0+2\chi_\gamma+2\theta_0\chi_\gamma)\theta_{\gamma,k} + (\eta k/\gamma)(\theta_0+2\chi_\gamma+2\theta_0\chi_\gamma).$

Figure 3.8.3. THE INFLOW AND OUTFLOW OF PROGRESSIONS FROM A TYPE


x
hit
miss
miss
hit
hit
miss

type is (1 0 0 1 1 0)

Now we have $\chi_\gamma \leq \theta\gamma^s$ and we take $\theta_0 \leq \theta\gamma^s$, where s is a constant to be chosen below. For θ sufficiently small we will have $\theta_0 \leq 1$ and so we can make the errors $\theta_{\gamma+\eta,k}$ only larger by writing

$$\text{(x)} \quad \theta_{\gamma+\eta,k} = 5\theta\gamma^{s-1}k\eta(1+\theta_{\gamma,k})+\theta_{\gamma,k}.$$

Going back through the above derivation and changing the signs of $\theta_{\gamma+\eta,k}$, $\theta_{\gamma,k}$, θ_0, and χ_γ gives us the error propagation equation for the negative errors. (Now we want all flows to be *minimal.*) We get the equivalent of (viii) for the absolute value of the error:

$$\begin{aligned}\theta_{\gamma+\eta,k} = \theta_{\gamma,k} &- (\eta l/\gamma)(\theta_0 + 2\chi_\gamma - 2\theta_0\chi_\gamma)\theta_{\gamma,k} \\ &+ (\eta l/\gamma)(\theta_0 + 2\chi_\gamma - 2\theta_0\chi_\gamma) \\ &+ [\eta(k-l)/(1-\gamma)](\theta_0 + 2\chi_\gamma - 2\theta_0\chi_\gamma)\theta_{\gamma,k} \\ &- [\eta(k-l)/(1-\gamma)](\theta_0 + 2\chi_\gamma - 2\theta_0\chi_\gamma).\end{aligned}$$

Since $\gamma \leq \alpha_0$ which we can take less than 1/2, we have $\gamma \leq 1-\gamma$, and so we can conclude that (ix) is valid for the absolute value of the negative errors as well. Thus equation (x) is justified for the absolute value of both positive and negative errors.

We still have to estimate the size of the flows that involve more than one of the ηm points. Let us look at an arithmetic progression of length k coming to x that changes type by receiving two of the ηm points. Suppose these two points occupy positions i and j of the progression respectively, $1 \leq i < j \leq k$. Let us fix the two types involved, which fixes i and j, and then ask how many progressions can flow between these two types. If the donating type is τ, then at most one such progression can flow for each pair (a,b) of the ηm points with the property that $a \in S(i,\tau,x,\Gamma)$ and a and b are in a distance ratio i:j from x. If we allow the ηm points to range over all m points, not just the remaining $(1-\gamma)m$ ones, we can only increase the flow in question. But now we are exactly in the situation covered by Theorem 3.4.4. and thus we can assert that our flow, except with exponentially small probability, will be $O(\eta|S(i,\tau,x,\Gamma)|m^{-\delta})$. Summing over all possible choices of i and j we still get a total possible inflow into the receiving type of $O(k_\alpha^2\gamma^{l-2}(1-\gamma)^{k-l+2}\eta m^{1-\delta})$ only, where l denotes the number of hits of the receiving type τ'. A trivial extension of the above argument shows that any flows into τ' arising from types with 3 (or 4, etc.) fewer hits will not be any greater. Thus the total magnitude of these flows combined will be $o(\gamma^l(1-\gamma)^{k-l}\eta m)$ and can therefore be incorporated into the relative errors permitted in equation (ix). Our derivation of equation (x) by ignoring type transitions which involve more than one of the ηm points has been justified.

We are finally at the point where we can push assertion (a) of our theorem through the induction step. We would like to prove

$$\theta_{\gamma+\eta,k} = (1+i_\theta)e^{5\theta[(\gamma+\eta)^s/s]k} - 1$$

$$= (1+i_\theta)e^{5\theta[(\gamma^s/s)+\eta\gamma^{s-1}+O(\eta^2)]k} - 1,$$

and since θ can be taken arbitrarily small, this is

$$(1+i_\theta)e^{5\theta(\gamma^s/s)k}(1+5\theta\gamma^{s-1}k\eta+O(k\eta^2)) - 1$$

$$= (1+\theta_{\gamma,k})(1+5\theta\gamma^{s-1}k\eta+O(k\eta^2)) - 1$$

$$= \theta_{\gamma,k}+5\theta\gamma^{s-1}k\eta(1+\theta_{\gamma,k})+(1+\theta_{\gamma,k})O(k\eta^2);$$

again the $O(k\eta^2)$ term is negligible compared to the $5\theta\gamma^{s-1}k\eta$ terms, and can be incorporated in the constant θ. Thus we need

$$\theta_{\gamma+\eta,k} = \theta_{\gamma,k} + 5\theta\gamma^{s-1}k\eta(1+\theta_{\gamma,k}),$$

which is exactly what we have proved in (x).

Next we prove (b) and determine the constant s. It will be simpler to let the sum

$$\chi_\gamma = \sum_{k=2}^{k_\alpha} \theta_{\gamma,k}\gamma^k$$

go to infinity, which we are allowed to do, since this can only increase the bound for χ_γ. So

$$\chi_\gamma \quad \le \quad \sum_{k=2}^{\infty} \theta_{\gamma,k}\gamma^k$$

then substituting $\theta_{\gamma,k}$ from (a) and letting

$$e^A = e^{5\theta(\gamma^s/s)},$$

we get

$$\begin{aligned}\chi_\gamma \quad &\le \quad (1+i_\theta)\gamma^2 e^{2A}/(1-\gamma e^A) - \gamma^2/(1-\gamma) \\ &= i_\theta\gamma^2 e^{2A}/(1-\gamma e^A) + (e^A-1)\gamma^2(1+e^A-\gamma e^A)/(1-\gamma)(1-\gamma e^A).\end{aligned}$$

For small θ we have $e^A \sim 1$, $e^A-1 \sim 5\theta(\gamma^s/s)$, and so if we take s so lage that

$$(5/s)\ (2\alpha^2/(1-\alpha)^2) < 1/2,$$

and i_θ so small that

$$i_\theta < ((1-\alpha)/2\alpha^2)\beta^s\theta$$

then we will have

$$\chi_\gamma \le \theta\gamma^s,$$

as desired. Note that s is independent of θ and γ.

The last object of interest is the residual sets. How fast can they grow? Let $r_\gamma^{(k,l)}$ denote $\rho(\tau,\gamma)m$ for τ a type of length k and l hits. The change in the r_γ's as we go from γ to $\gamma+\eta$ load factor can be computed in a manner analogous to the above. The maximum flow into τ during the current step will be

$$l r_\gamma^{(k,l-1)}\ \eta(1+\theta)(1+2\chi_\gamma)/(1-\gamma) \ \le\ 2l r_\gamma^{(k,l-1)}\eta/(1-\gamma).$$

We are not counting the outflows, so we obtain the recurrence relation

$$r_{\gamma+\eta}^{(k,l)} = r_{\gamma}^{(k,l)} + 2\,l r_{\gamma}^{(k,l-1)}\eta/(1-\gamma), \qquad 0 \leq l \leq k\ ,$$
$$(r_{\beta}^{(k,l)} = E_0).$$

(Here we see that types with more hits will grow faster than types with fewer, since they have more types feeding into them. The same, of course, was true in our computation of the relative errors, but there we decided to ignore this improvement. Because the residual sets cause the argument to fail for large α, we want to do a better job of estimating their growth.) Since all initial values are identical, it is clear from (xi) that $r_{\gamma}^{(k_\alpha,k_\alpha)}$ is the maximally growing type. In what follows therefore we restrict ourselves to estimating its growth. Again, since η is infinitesimal compared to l or r_γ, we can easily check that the solution to the above difference equations (which we can think of as the system of differential equations

$$dr^{(k,l)}/d\gamma = 2lr^{(k,l-1)}/(1-\gamma),\ 0 \leq l \leq k)$$

is of the form

$$r_{\gamma}^{(k,l)} = E_0\ (1 + 2\log\ (1/(1-\gamma)) - 2\log\ (1/(1-\beta)))^{l}$$
$$\leq E_0\ (1 + 2\log\ (1/(1-\gamma)))^{l}.$$

Thus

$$r_{\gamma}^{(k_\alpha,k_\alpha)} \leq E_0\ [1 + 2\log(1/(1-\gamma))]^{(1/2+\delta_3)\log m/\log(1/\alpha)}$$
$$= E_0\ m^{(1/2+\delta_3)\log[1+2\log(1/(1-\gamma))]/\log(1/\alpha)},$$

and so

$$\rho_\gamma \leq \left(E_0 m^{(1/2+\delta_3)\ \log[1+2\log(1/(1-\gamma))]/\log(1/\alpha)}\right)/m,$$

as (c) of Theorem 3.8.1 requires.

There are at most $1/\eta = m^{1/4+\delta_1}$ steps, at most m points x, at most $\Sigma_{2\leq k\leq k_\alpha} 2^{k_\alpha} \leq 2^{k_\alpha+1} = 2m^{(1/2+\delta_3)\log 2/\log(1/\alpha)}$ distinct types and at most k_α values for i in the context $S(i,\tau,x,\Gamma)$. Thus the total number of excluded cases is a polynomial in m, and no case has probability higher than $\exp(-C_\theta m^{\delta_4})$. Thus as m gets large the total of the probabilities of the excluded cases is less than $\exp(-C_\theta m^{\delta_0})$, where δ_0 is as constrained in (ii).

This completes our proof of Theorem 3.8.1. ∎

COROLLARY 3.8.1. Given $0<\beta<\alpha<\alpha_0<1$, for any small positive constant θ, there exists an initial configuration of βm occupied points, such that if we add $(\alpha-\beta)m$ additional points using the double hashing process, then except with probability $\exp(-C_\theta m^{\delta_0})$, we will arrive at a configuration of αm occupied positions such that for each point x and for each length k, $2\leq k\leq k'_\alpha$, the number of arithmetic progressions coming to x of length k from the occupied points will be $\alpha^k(1\pm\underline{\theta}_{\alpha,k})$. Here the $\underline{\theta}_{\alpha,k}$ are relative errors satisfying

$$\text{(a)} \qquad \sum_{k=2}^{k'_\alpha} \underline{\theta}_{\alpha,k}\alpha^k \leq \theta, \text{ and}$$

$$\text{(b)} \qquad \alpha^{k'_\alpha}(1\pm\underline{\theta}_{\alpha,k'_\alpha})m \leq m^{1/2-\delta'},$$

for some positive constants δ_0, δ'. (Notice that we have excluded any references to bad insertions or residual progressions.)

Proof. This is a direct consequence of Theorem 3.8.1, except for a few items that we need to check. First is the existence of a good initial configuration, the "seed" of the extension process. We have to choose a configuration of βm points so that for each point x and each type τ of length k and l hits we have

$$\beta^l(1-\beta)^{k-l}m(1\pm i_\theta)$$

progressions of type τ coming to x, with i_θ, k, l restricted as in the theorem. Now if $\beta^l(1-\beta)^{k-l}m\geq m^{1/4}$, then by Theorem 3.4.2., all configurations except for a fraction not exceeding $\exp(-Ci_\theta^2 m^{1/5})$ of them will satisfy the above condition. If $\beta^l(1-\beta)^{k-l}m \ll m^{1/4}$, then let τ_1 denote a shorter type which is an initial segment of τ, of length k_1 and l_1 hits, such that $\beta^{l_1}(1-\beta)^{k_1-l_1} m \approx m^{1/4}$. Then we can apply Corollary 3.4.2 to τ_1 and claim it has at most $O(m^{1/4})$ progressions. Clearly τ cannot have more progressions than τ_1, so therefore the excess of progressions τ can have is at most $O(m^{1/4})$. Any such excessive progressions we label as residual for out type τ. This is consistent with the assumptions of Theorem 3.8.1. that allow initial residual errors as large as $m^{1/4+\delta_2}$ per type. Therefore all configurations of the βm points except for an exponentially small fraction of them satisfy the initial conditions of Theorem 3.8.1. We start the extension process by choosing one of them. This is an interesting "non-constructive" aspect of our proof. We do not know how to find a specific such good configuration, though we have just proved that almost all configurations are good.

We now perform the extension process till we reach the load factor α, as described in Theorem 3.8.1. The number of points inserted with bad insertions is $O(m^{1/2-\delta})$. For each point x and each length k, no bad point can introduce (influence) more than k progressions of length k coming to x. Thus the bad points can introduce at most $O(km^{1/2-\delta})$ progressions of length k coming to x, $k \leq k_\alpha = O(\log m)$. The number of residual progressions of length k coming to x is at most

$$E_0 m^{(1/2+\delta_3)\log[1+2\log(1/(1-\alpha))]/\log(1/\alpha)}$$

$$= m^{1/4+\delta_2+(1/2+\delta_3)\log[1+2\log(1/(1-\alpha))]/\log(1/\alpha)}.$$

The absolute constant α_0 is chosen so that

$$1/4 \ + \ \delta_2 \ + \ (1/2+\delta_3)\log[1+2\log(1/(1-\alpha))]/\log(1/\alpha) \quad \leq \quad 1/2 \ - \ \delta$$

for $\alpha \leq \alpha_0$. A rough numerical computation shows that

$$\alpha_0 \sim .319.$$

Thus the contribution of the residual progressions at any length is at most $O(m^{1/2-\delta})$. Let now δ', δ'' be such that $0<\delta'<\delta''<\delta,\delta_3$. Let $k'_\alpha < k_\alpha$ be such that

$$\alpha^{k'_\alpha} m = m^{1/2-\delta''};$$

such a k'_α clearly exists since

$$\alpha^{k_\alpha} m = m^{1/2-\delta_3}.$$

We have

$$1+\theta_{\alpha,k'_\alpha} = (1+i_\theta)e^{5\theta(\alpha^s/s)k'_\alpha} =$$

$$= (1+i_\theta)m^{5\theta(\alpha^s/s)(1/2+\delta'')/\log(1/\alpha)}.$$

Recall that $i_\theta \to 0$ as $\theta \to 0$, and so by choosing θ sufficiently small we can obtain

$$1+\theta_{\alpha,k'_\alpha} \leq m^{\delta'''}, \text{ for } \delta'''<\delta''-\delta'.$$

So we have

$$\alpha^{k'_\alpha} m(1+\theta_{\alpha,k'_\alpha}) \leq m^{½-\delta''+\delta'''}.$$

We can add to this the contribution of the bad insertions and the residual progressions, and since they both are $O(m^{1/2-\delta''})$, the grand total of progressions of length k'_α coming to x is

$$\alpha^{k'_\alpha} m(1+\underline{\theta}_{\alpha,k'_\alpha}) \leq m^{½-\delta'}.$$

This proves part (b) of the Corollary.

For part (a) we work analogously. We know that

$$\text{(xii)} \qquad \sum_{k=2}^{k_\alpha} \theta_{\alpha,k}\alpha^k \leq \theta\alpha^s \leq \theta.$$

The contributions of the bad insertions and the residual progressions estimated as above are $O(m^{1/2-\delta''})$ even when summed over all allowed lengths k. Thus these contributions to (xii) can be incorporated in the constant θ. Since $k'_\alpha \leq k_\alpha$, we have shown that the *true* relative errors satisfy

$$\sum_{k=2}^{k'_\alpha} \underline{\theta}_{\alpha,k}\alpha^k \leq \theta$$

as desired.

By Theorem 3.8.1. the probability of the excluded events is $\exp(-C_{\theta} m^{\delta_0})$. This completes the argument. ∎

Remark. It is worth pointing out the reason why we have carried out the computation of the growth of the residual progressions separately from the regular ones. For the regular progressions, the initial number of progressions of a type τ of length k and l hits is approximately $\beta^{l}(1-\beta)^{k-l}m$. Thus for β small and a particular k we have most regular progressions in types with few hits. The initial number of residual progressions, however, is the same for all types, thus giving rise to a quantitatively different model.

Figure 3.8.4. is to be used for reference. It summarizes the various δ's we have introduced and the relations among them.

DEFINITIONS.

$\eta = m^{-1/4-\delta_1}$

$\beta^{k_\beta m} = m^{1/4+\delta_2}$

$\alpha^{k_\alpha m} = m^{1/2-\delta_3}$

$E_0 = m^{1/4+\delta_2}$

prb. of excluded events $= e^{-C_\theta m \delta_0}$

Prb. of bad insertion $= m^{-1/2-\delta}$

CONSTRAINTS.

$\delta_2 > \delta_1 > 0$

$\delta_2 - \delta_1 > \delta_4 > \delta_0 > 0$

$2\delta_1, \delta_3 > \delta_6 > \delta > 0$

$2\delta_1 - \delta, \delta_3 - \delta > \delta_5 > 0$

$\delta_3, \delta > \delta'' > \delta' > 0$

Figure 3.8.4. THE PROLIFERATION OF DELTAS

Appendix

On the Distribution of Arithmetic Progressions of Length 3 in a Random Sample of Integers 1,2,...,N -- A Derivation Using the Exponential Sums Technique

We consider subsets S of $[1,N]$ generated by the following random process: each $x \in [1,N]$ is chosen to belong to S with a fixed probability λ, $0<\lambda<1$, independently of all the other x.

[*Caveat*: We have chosen this probability model versus the one in which we regard all S with $|S| = \lambda N$ as equally likely because the manipulations are easier; almost certainly the identical derivation holds for this second model -- places in the argument where the two models differ are indicated by ‡.]

If we consider the exponential sum ($\alpha \in R$)

$$S(\alpha) = \sum e(\alpha x), \text{ (we write } e(t) \text{ for } e^{2\pi it}),$$

then it is easy to see that the number of arithmetic progressions of length 3 in S is given by

$$\int_0^1 S^2(\alpha)S(-2\alpha)d\alpha.$$

If we set $T(\alpha) = \lambda \, \Sigma \, e(\alpha x)$, then it is again easy to see that the *average* number of arithmetic progressions of length 3 of a set S generated as above is

$$(‡) \qquad \int_0^1 T^2(\alpha)T(-2\alpha)d\alpha = \lambda^3N^2/4 + O(N) \text{ as } N \to \infty.$$

The constants implied in all our uses of the *O* notation will be absolute.

A set S generated by the above process will be called (m,ε)-equidistributed, $\varepsilon \ll \min\{\lambda,1-\lambda\}$, $m \leq N$, if for all q, $1 \leq q \leq N^{1/2}$, and for all n, $1 \leq n \leq N-mq$, the number of elements of S that lie on the arithmetic progression $n, n+q, n+2q, \ldots, n+(m-1)q$, is not more than $(\lambda+\varepsilon)m$, and furthermore $|S| \geq (\lambda-\varepsilon)N$.

In order to study the deviations of the number of arithmetic progressions of length 3 from the average, we need the following three fundamental lemmas.

The first lemma is just a summary of trivial but useful properties.

LEMMA 1. We have, with S,T as above

(1) $S(\alpha)$, $T(\alpha)$ are periodic of period 1;

(2) $S(\alpha) = O(\lambda N)$, $T(\alpha) = O(\lambda N)$ as $N \to \infty$;

(3) $\int_0^1 |S(\alpha)|^2 d\alpha = \lambda N$;

(4) $T(\alpha) = O(\lambda/\|\alpha\|)$, as $\|\alpha\| \to 0$, where $\|\alpha\|$ denotes the distance from α to the nearest integer. ▮

The next lemma asserts that for equidistributed sets S, $S(\alpha)$ is well approximated by $T(\alpha)$.

LEMMA 2. If a set S generated by the above process is (m,ε)-equidistributed, then for $m = o(N^{1/2})$

(5) $|S(\alpha)-T(\alpha)| \leq 3\varepsilon N+O(mN^{1/2})$. ▮

The last lemma shows that non-equidistributed sets are extremely rare.

LEMMA 3. The probability that a set S generated by the above process *is not* (m,ε) -equidistributed is

$O(N^{3/2}\, e^{-\varepsilon^2 m})$, as $N,m \to \infty$ ▮

Using those three lemmas we can then prove our main result, which is

THEOREM. With probability $1-O(N^{3/2}\, e^{-\varepsilon^2 m})$ we have

$$\left|\int_0^1 S^2(\alpha)S(-2\alpha)d\alpha - \int_0^1 T^2(\alpha)T(-2\alpha)d\alpha\right| \leq$$

$$3\varepsilon\lambda N^2 + O(\lambda m N^{3/2} + \lambda^2(\lambda+\varepsilon)N^{9/5}), \text{ as } N,m \to \infty. ▮$$

Lemma 1 is obvious except possibly for part (4). This is easily done by summing the geometric series involved.

We now make two comments and then proceed to prove Lemmas 2 and 3 and the Theorem.

Comment 1. From the proof of Lemma 2 it will be clear that we could have allowed up to $O(mN^{1/2})$ "exceptions" (i.e., selections of arithmetic progressions not satisfying the stated conditions) and still gotten our result. This almost certainly implies that in our proof of Lemma 3 we can get a sharper estimate, possibly $O(N^{3/2}\ e^{-\varepsilon^2 m^2})$.

Comment 2. The optimal value of m for the theorem is to choose m as large as possible while still consistent with the assumption $m = O(N^{1/2})$. The theorem then, loosely speaking, states that for any small $\delta>0$, with probability $1-O(e^{-\mu(\varepsilon)N^{1/2-\delta}})$, where $\mu(\varepsilon)$ is some function of $\varepsilon>0$, the number of arithmetic progressions of length 3 in a set S generated by the above process will lie in the interval $[((\lambda^3/4) - \varepsilon)N^2,\ ((\lambda^3/4) + \varepsilon)N^2]$, as $N \to \infty$.

Proof of Lemma 2. By looking at the continued fraction expansion of α, we can find h,q,β such that

$$\alpha = (h/q) + \beta,\ (h,q) = 1,\ q \leq N^{1/2},\ q|\beta| \leq 1/N^{1/2}.$$

We now start from the relation

$$S(\alpha) = (1/mq) \sum_{r=1}^{q} \sum_{n=1}^{m} \sum_{\substack{n \leq x < n+mq \\ x \in S,\ x \equiv r (\mathrm{mod}\ q)}} e(\alpha x) + O(mq);$$

this relation is true because, for given x,m,q, there are exactly mq integers n satisfying

$$n \leq x < n+mq,$$

and these integers n also satisfy $1 \leq n \leq N$ provided that $mq \leq x < N-mq$. Thus the coefficient of $e(\alpha x)$ on the RHS is unity except when $x<mq$ or $x \geq N-mq$, these cases being compensated for by the error term.

We also have $e(\alpha x) = e(rh/q)e(\beta n) + O(mq|\beta|)$. For each r,n the number of terms in the inner sum is at most $(\lambda+\varepsilon)m$ by our assumption of (m,ε)-equidistribution, thus it is $(\lambda+\varepsilon)m - D(m,n,q,r)$, where $D \geq 0$.

Therefore

$$\text{(a)} \qquad S(\alpha) = (\lambda+\varepsilon)\ (1/q) \sum_{r=1}^{q} e(rh/q) \sum_{n=1}^{N} e(\beta n)\ -$$

$$-(1/mq) \sum_{r=1}^{q} e(rh/q) \sum_{n=1}^{M} e(\beta n)D(m,n,q,r) + O(mq) + O(Mmq|\beta|).$$

If we put $\beta=0$ and $h=0$ (legitimate since we have not yet used $(h,q) = 1$) in the above we get

$$\text{(b)} \qquad |S| = (\lambda+\varepsilon)N - (1/mq) \sum_{r=1}^{q} \sum_{n=1}^{M} D(m,n,q,r) + O(mq).$$

Since S is equidistributed we have $(\lambda-\varepsilon)N \leq |S| \leq (\lambda+\varepsilon)N$; using this and the facts that $q \leq N^{1/2}$, $q|\beta| \leq 1/N^{1/2}$, we can combine (a) and (b) into

$$\left|S(\alpha) - (\lambda/q) \sum_{r=1}^{q} e(rh/q) \sum_{n=1}^{N} e(\beta n)\right| \leq 3\varepsilon N + O(mN^{1/2}).$$

We now distinguish two cases. If $\|\alpha\| \leq 1/N^{1/2}$, then we have $h=0$, $q=1$, $\beta=\alpha$ and the above relation becomes

$$|S(\alpha) - T(\alpha)| \leq 3\varepsilon N + O(mN^{1/2})$$

which is what we want. If however $\|\alpha\| > 1/N^{1/2}$, then we cannot have $q=1$. But in the event $q>1$,

$$\sum_{r=1}^{q} e(rh/q) = 0,$$

so the above relation becomes $|S(\alpha)| \leq 3\varepsilon N + O(mN^{1/2})$. But from Lemma 1, (4) we have

$$T(\alpha) = O(\lambda/\|\alpha\|) = O(\lambda N^{1/2}),$$

and therefore again

$$|S(\alpha)-T(\alpha)| \leq 3\varepsilon N + O(mN^{1/2}).$$

Q.E.D. ▮

Proof of Lemma 3. We use the following fundamental property of the binomial distribution:

$$(*) \qquad \sum C(a,t)p^a q^{a-t} \leq \exp[(a-k)\ \log(aq/(a-k)) + k\ \log(ap/k)]$$

where the sum is either over t such that $t \geq k$, or over t such that $t \leq k$, provided that $k \geq pa$ or $k \leq pa$ respectively.

For S *not* to be equidistributed we must have

(a) $|S| \leq (\lambda-\varepsilon)N$ or

(b) $\exists$ n,q s.t. the number of elements of S in the progression $n, n+q, \ldots, n+(m-1)q$ is $>(\lambda+\varepsilon)m$.

Using the elementary fact that

$$[(1-\lambda\pm\varepsilon)\ \log(1-\lambda)/(1-\lambda\pm\varepsilon) + (\lambda\mp\varepsilon)\ \log(\lambda/(x\mp\varepsilon))] \leq -\varepsilon^2$$

we see that the probability of (a) happening is $O(e^{-\varepsilon^2 N})$ (set $k = (\lambda-\varepsilon)N$, $a=N$, $p=\lambda$, $q=1-\lambda$ in (*)), and the probability of (b) happening is certainly less than

$$\sum O(e^{-\varepsilon^2 m}) = O(N^{3/2}\ e^{-\varepsilon^2 m})$$

(set $k = (\lambda+\varepsilon)m$, $a=m$, $p=\lambda$, $q=1-\lambda$ in (*)).

So the probability of either (a) or (b) happening is

$$O(N^{3/2}\ e^{-\varepsilon^2 m}) \text{ as } N \to \infty,\ m \to \infty,$$

as the lemma asserts. ▮

Proof of the Theorem. (For *O* notation: $N \to \infty$, $\delta \to 0$.) By Lemma 3 we may assume that S is (m,ε) -equidistributed. We have

$$|S^2(\alpha)S(-2\alpha) - T^2(\alpha)T(-2\alpha)| =$$

$$|(S^2(\alpha)-T^2(\alpha))S(-2\alpha) + T^2(\alpha)(S(-2\alpha)-T(-2\alpha))| \leq$$

$$|S(-2\alpha)(S(\alpha)+T(\alpha))(S(\alpha)-T(\alpha))| + |T^2(\alpha)(S(-2\alpha)-T(-2\alpha))|$$

and by Lemma 2 and Lemma 1, (2) it follows that

$$(6)\quad |S^2(\alpha)S(-2\alpha) - T^2(\alpha)T(-2\alpha)| = O(\varepsilon\lambda^2N^3).$$

From Lemma 2 and Lemma 1, (4), we have

$$|S(\alpha)|\leq 3\varepsilon N+O(\lambda/\|\alpha\|+mN^{1/2}),$$

and thus

$$\int_\delta^{1/2}|S(\alpha)|d\alpha \leq 3\varepsilon N/2 + O(-\lambda\log\delta + mN^{1/2}),$$

which, together with Lemma 1, (3), implies that

$$(7)\quad \left|\int_\delta^{1-\delta} S^2(\alpha)S(-2\alpha)d\alpha\right| \leq 3\varepsilon\lambda N^2 + O(-\lambda^2N\log\delta + \lambda mN^{3/2}).$$

By (6), Lemma 1, (2), we also have

$$(8)\quad \int_{-\delta}^{\delta} S^2(\alpha)S(-2\alpha)d\alpha = \int_{-\delta}^{\delta} T^2(\alpha)T(-2\alpha)d\alpha + O(\varepsilon\lambda^2N^3\delta).$$

Finally, from Lemma 1, (4) and Cauchy-Schwartz we have

$$\left|\int_\delta^{1-\delta} T^2(\alpha)T(-2\alpha)d\alpha\right|^2 \leq \left(\int_\delta^{1-\delta}|T(\alpha)|^2d\alpha\right)^2 \int_\delta^{1-\delta}|T(-2\alpha)|d\alpha =$$

$$\left(\int_\delta^{1-\delta} O(\lambda^2/\|\alpha\|^2)d\alpha\right)^3 = O(\lambda^6/\delta^3) \text{ as } \delta \to 0.$$

So

$$(9)\quad \left|\int_\delta^{1-\delta} T^2(\alpha)T(-2\alpha)d\alpha\right| = O(\lambda^3\delta^{-3/2}),\ \delta \to 0.$$

We now write

$$|\int_0^1 S^2(\alpha)S(-2\alpha)d\alpha - \int_0^1 T^2(\alpha)(-2\alpha)d\alpha| \le$$

$$|\int_{-\delta}^{\delta} S^2(\alpha)S(-2\alpha)d\alpha - \int_{-\delta}^{\delta} T^2(\alpha)T(-2\alpha)d\alpha| +$$

$$|\int_{\delta}^{1-\delta} S^2(\alpha)S(-2\alpha)d\alpha| + |\int_{\delta}^{1-\delta} T^2(\alpha)T(-2\alpha)d\alpha| \le$$

(by (7), (8), (9))

$$\le 3\varepsilon\lambda N^2 + O(\lambda^2 N\log\delta + \lambda m N^{3/2} + \varepsilon\lambda^2 N^3\delta + \lambda^3\delta^{-3/2}) .$$

Setting $\delta = N^{-1-1/5}$ in the above to get minimum growth of the O term, we obtain the desired result:

If S is (m,ε)-equidistributed, then

$$|\int_0^1 S^2(\alpha)S(-2\alpha)d\alpha - \int_0^1 T^2(\alpha)T(-2\alpha)d\alpha| \le 3\varepsilon\lambda N^2 + O(\lambda m N^{3/2} + \lambda^2(\lambda+\varepsilon)N^{9/5}).$$

∎.

The arguments in this appendix were motivated by the discussion of [Roth].

REFERENCES

[Bel-Kam]
Bell, J. R. and Kaman, C. H., **"The Linear Quotient Hash Code"**, CACM 13, 11 (November 1970), 675-677

[Berge]
Berge, Claude, Principles of Combinatorics, Chapter 1, Academic Press, 1971

[Bobrow]
Bobrow, Daniel G., **"Combined Open Address and Chained Hash Search"**, Internal Memo, Xerox Palo Alto Research Center, January 1974

[Brent]
Brent, Richard P., **"Reducing the Retrieval Time of Scatter Storage Techniques"**, CACM 16, 2 (February 1972), 105-109

[Chernoff]
Chernoff, H., **"A Measure of Asymptotic Efficiency for Tests of Hypotheses Based on a Sum of Observations"**, Ann. Math. Stat. 23, 493-509

[Erd-Sp]
Erdos, Paul and Spencer, Joel, Probabilistic Methods in Combinatorics, Academic Press, New York, 1974

[Feller]
Feller, William, An Introduction to Probability Theory and its Applications, Vol. 1, Third Edition, Section II.6, John Wiley and Sons, 1968

[Hardy]
Hardy, G. H. and Wright, E. M., An Introduction to the Theory of Numbers, Fourth Edition, Chapters III, X, Oxford, 1968

[Knuth1]
Knuth, Donald E., The Art of Computer Programming, Vol. 1, (Fundamental Algorithms) Section 6.4, Addison-Wesley, Second Edition, 1975

[Knuth2]
Knuth, Donald E., The Art of Computer Programming, Vol. 3, (Sorting and Searching) Section 1.2, Addison-Wesley, 1973

[Knuth3]
Knuth, Donald E., **"Mathematical Analysis of Algorithms"**, Information Processing Letters, 1971, North-Holland (1972), 19-27

[Liu]
Liu, C. L., **Introduction to Combinatorial Mathematics**, Chapter II, Mc-Graw Hill, 1968

[Renyi]
Renyi, Alfred, **Probability Theory**, Chapter VII, North Holland, 1970

[Paterson}
Paterson, Michael, **Personal Communication**, 1976

[Roth]
Roth, Klaus, F., **"On Certain Sets of Integers"**, J. London Math. Soc. 28, (1953), 104-109

[Ullman]
Ullman, J. D., **"A Note on the Efficiency of Hash Functions"**, JACM 19 (1972), 569-575

INDEX

PAGE

algorithm A (bucket search) 6
algorithm C (chained hashing with coalescing lists) 7
algorithm D (double hashing) 5,83
algorithm L (linear probing) 4
algorithmic analysis 8
arithmetic progression 4,10,56ff

bad insertion 99

chained hash algorithm 6
clustering 10
collision 2
collision resolution strategy 2,4
comparison 9
controlling the intersections 94

double hashing 4,83

excessive flow 94
exponential generating function 31
extension process 93

Farey series 65
fixpoint 66
flow 93

generating function 15,21

hash function 2,4
hit 56
hypergeometric distribution 61

key 2

linear probing 4
link 6

miss	61
open addressing	4
overflow	2
record	2
residual progression	97
recurrence relation	21
type (of a progression)	56
secondary clustering	11
tertiary clustering	11
uniform hashing (or probing)	11